VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

# DIPLOMOVÁ PRÁCE

Brno, 2019 Bc. Tomáš Caha


# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS


# ANALÝZA ZDROJŮ NEVYŽÁDANÉ ELEKTRONICKÉ POŠTY
ANALYSIS OF EMAIL SPAM SOURCES

**DIPLOMOVÁ PRÁCE**
MASTER'S THESIS


**AUTOR PRÁCE** — **Bc. Tomáš Caha**
AUTHOR

**VEDOUCÍ PRÁCE** — **doc. Ing. Dan Komosný, Ph.D.**
SUPERVISOR


**BRNO 2019**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

***Student:*** Bc. Tomáš Caha ***ID:*** 164249
***Ročník:*** 2 ***Akademický rok:*** 2018/19

**NÁZEV TÉMATU:**

## Analýza zdrojů nevyžádané elektronické pošty

**POKYNY PRO VYPRACOVÁNÍ:**

Popište možnosti nalezení polohy zařízení podle IP adresy. Vytvořte aplikaci, která bude provádět odhad geografické polohy pro zadanou IP adresu. K tomu účelu využijte volně dostupné i placené geolokační databáze s omezeným počtem dotazů zdarma. Následně proveďte odhad polohy pro IP adresy, které jsou uvedené ve zvolených volně dostupných seznamech. Zaměřte se na seznamy, které uvádí IP adresy zdrojů nevyžádané pošty nebo kybernetických útoků. Proveďte analýzu získaných výsledků. Aplikaci sestavte v programovacím jazyce Python 3. Vytvořený kód vystavte pod licencí MIT a umístěte jej na repositář PyPI.

**DOPORUČENÁ LITERATURA:**

[1] PUŽMANOVÁ, R. TCP/IP v kostce. 2. vyd. Kopp, 2009. 620 s. ISBN: 978-80-7232-388-3.

[2] PILGRIM, M. Ponořme se do Python(u) 3. CZ.NIC, 2010. 435 s. ISBN: 978-80-904248-2-1.

***Termín zadání:*** 1.2.2019 ***Termín odevzdání:*** 16.5.2019

***Vedoucí práce:*** doc. Ing. Dan Komosný, Ph.D.
***Konzultant:***

**prof. Ing. Jiří Mišurec, CSc.**
*předseda oborové rady*

## ABSTRAKT

První část této práce se věnuje problematice návrhu a vytvoření aplikace, která provádí odhad geografické polohy pro zadanou IP adresu za využití volně dostupných i komerčních geolokačních databází. Jsou shrnuty používané metody geografické lokalizace síťových zařízení a množství údajů poskytovaných vybranými volně dostupnými i komerčními geolokačními databázemi. Především jsou popsány způsoby získávání informací o IP adresách z jednotlivých databází. V práci jsou představeny způsoby, jakých bylo využito při vytváření aplikace v jazyce Python a jejích dílčích součástí, které lze jednoduše znovu použít v jiných programech. Dílčí části aplikace také tvoří jeden funkční celek obsahující spustitelný program. Celá aplikace je volně dostupná pod licencí MIT a zveřejněna na GitHubu (`https://github.com/tomas-net/ip2geotools/`) a ve veřejném seznamu balíčků open-source knihoven pro Python zvaném PyPi. Druhá část této práce se věnuje popisu kybernetických hrozeb a použití vytvořené aplikace k hromadnému odhadu geografické polohy IP adres, které jsou uvedené ve vybraných volně dostupných seznamech jako zdroje nevyžádané pošty nebo kybernetických útoků. Geografická analýza zdrojů kybernetických hrozeb popisuje, ze kterých zemí k útokům dochází. Výsledky analýzy jsou prezentovány ve formě grafů a map.



## KLÍČOVÁ SLOVA

aplikace, databáze, geolokace, IP adresa, ip2geotools, nevyžádaná pošta, Python



## ABSTRACT

The first part of this thesis deals with an approach of designing and developing of application for IP geolocation using various geolocation databases. Methods of geographical location of network devices and amount of available data provided by chosen commercial and freely accessible geolocation databases are presented. The data are summarized with focus on methods of obtaining information about IP addresses from various databases. In the paper there are also presented ways used to develop the Python application and its parts, which can be easily reused in other programs. The command line program was created to demonstrate that all parts of the developed application work properly. The whole application is freely accesible under the conditions of the MIT license, published on GitHub (`https://github.com/tomas-net/ip2geotools/`) and in the Python package index PyPi. The second part of this thesis deals with the description of cyberthreats and the use of developed application to mass geolocation of IP addresses that are listed in chosen freely accessible lists as sources of email spam or cyberattacks. Geographical analysis of cyberattacks sources shows countries of origin. Results of analysis are presented in graphs and maps.



## KEYWORDS

application, database, geolocation, IP address, ip2geotools, Python, spam

CAHA, Tomáš. *Analýza zdrojů nevyžádané elektronické pošty*. Brno, 2019, 100 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Dan Komosný, Ph.D.

# PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Analýza zdrojů nevyžádané elektronické pošty“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ..... .....

podpis autora

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Danu Komosnému, Ph.D. za odborné vedení, konzultace a podnětné návrhy k práci.


Brno ..... .....

podpis autora

# Obsah

# Seznam obrázků

# Seznam výpisů

# Úvod

Mým úkolem je seznámit se s možnostmi nalezení polohy zařízení se známou IP adresou. Způsoby, kterými lze polohu síťového zařízení odhadovat, jsou popsány v kapitole 1. Jedním z cílů této práce je vytvořit aplikaci, která bude provádět odhad geografické polohy pro zadanou IP adresu s využitím geolokačních databází, které jsou detailně popsány v kapitole 2. Pozornost je věnována možnostem jednotlivých databází a popisu jejich API (rozhraní pro programování aplikací – Application Programming Interface). Dalším z mých úkolů je seznámit se s druhy kybernetických hrozeb, což je přiblíženo v kapitole 3. V kapitole 4 se lze dočíst o návrhu celé aplikace a jednotlivých částech, které jsou zveřejněné na GitHubu pod licencí MIT a ve veřejném seznamu balíčků open-source knihoven pro Python zvaném PyPi. Kapitola 5 je věnována analýze zdrojů kybernetických útoků, resp. geolokaci IP adres, které jsou uvedené ve volně dostupných seznamech evidujících IP adresy s podezřelou aktivitou. Výsledky analýzy původu nevyžádané pošty, nevyžádaných příspěvků v diskuzních fórech, škodlivého softwaru, vyděračského softwaru, útoků na redakční systém WordPress a výstupních uzlů sítě Tor jsou prezentovány ve formě grafů a map. Analýze IP adres, ze kterých států pocházejí různé druhy kybernetických útoků, resp. které státy jsou do těchto hrozeb zapojeny, se zabývá poměrně málo odborných publikací a článků, což činí tuto práci neobvyklou. Závěrem jsou diskutovány výhody mé aplikace a výsledky analýzy zdrojů kybernetických útoků.

# 1 Geografická lokalizace síťových zařízení

Geografická lokalizace je v dnešní době Internetu důležitá. Jedná se o proces zjišťování fyzické polohy síťového zařízení, resp. uživatele na základě vybraných síťových parametrů. Jelikož Internet na rozdíl od států nemá žádné pevně stanovené hranice, je velmi složité zjistit pozici koncového zařízení připojeného k internetu[1].

## 1.1 Využití geolokace

Geolokace má nepřeberné množství využití, které bych rozdělil do dvou významných skupin, a to na využití geolokace pro zvýšení komfortu uživatele a na bezpečnost, resp. ochranu uživatele.

### 1.1.1 Zvýšení komfortu uživatele

Do této skupiny je možné zařadit využití geolokace v aplikacích, kde je z hlediska uživatelského komfortu potřebné nebo výhodné znát fyzickou polohu uživatele. Díky znalosti geografické polohy uživatele jsme schopni přesně cílit reklamu (např. nabízet obchody a jejich služby nebo produkty v blízkém okolí). Na internetových stránkách se zprávami, s kulturními akcemi nebo předpověďmi počasí můžeme automaticky nabízet lokalitu, ve které se uživatel právě nachází, resp. můžeme uživateli automaticky zasílat informace z okolí, kde se právě nachází.

### 1.1.2 Bezpečnost a ochrana uživatele

Druhou, neméně důležitou skupinou využití geolokace je bezpečnost, resp. ochrana uživatele. Internet nezná hranic, právní postižitelnost kyberkriminality je stále velkou neznámou. Uživatelé mohou mezi sebou komunikovat na velké vzdálenosti a toho je možné zneužít. Fyzická poloha uživatele, resp. útočníka může v takovém případě napomoci k jeho dopadení nebo alespoň k omezení síťového provozu z rizikových lokalit. Informace o fyzické poloze uživatele, resp. síťového zařízení může taktéž mít preventivní účinek, např. vyžádání dvoufaktorové autentizace při přihlašování z oblasti, kde se uživatel většinou nenachází.

## 1.2 Metody geografické lokalizace síťových zařízení

Metod pro geografickou lokalizaci známe hned několik. Dělíme je do dvou základních skupin: pasivní a aktivní metody. V mé práci se v podstatě využívá pasivní metody geolokace na základě IP adresy síťového zařízení.

### 1.2.1 Pasivní metody

Pasivní metody využívají především informací o síťovém zařízení uložených v komerčních nebo veřejných geolokačních databázích, které většinou obsahují bloky IP adres namapované na geografickou polohu[2]. Pasivní metody lze dále dělit podle toho, jakých údajů využívají, a to na geolokaci podle:

- IP adresy,
- DNS (hiearchický systém doménových jmen – Domain Name System),
- seznamu dostupných bezdrátových sítí Wi-Fi (sada standardů popisujících bezdrátový přenos dat v počítačových sítích – Wireless Fidelity).

#### Geolokace podle IP adresy

Každé zařízení připojené k počítačové síti, resp. k internetu má IP adresu. Známe-li IP adresu, jsme schopni o ní informace vyhledat v různých geolokačních databázích nebo v databázi organizace pro přidělování adres IANA (Internet Assigned Numbers Authority), resp. v databázích koordinačních středisek, které pod tuto organizaci spadají. Jednotlivé databáze poskytují různé množství informací s různou přesností a aktuálností informací.

#### Geolokace podle DNS

Určování polohy síťového zařízení ze znalosti jeho DNS záznamu nám dá pouze velmi hrubou a v mnoha případech nepřesnou představu o tom, kde se zařízení nachází. Využijeme-li reverzního DNS záznamu, kdy se nám IP adresa přeloží na doménové jméno, vzhledem k povaze doménových jmen, která jsou strukturovaná a hierarchická, zjistíme například dle národní domény 1. řádu přibližnou zemi, kde by se zařízení mohlo nacházet. Je nutné upozornit na to, že zařízení s určitou IP adresou a doménovým jménem, jehož doména 1. řádu je například .CZ, vůbec nemusí být umístěno v České republice, ale kdekoliv na světě. Pokud doménové jméno nepatří pod národní domény, ale pod generické, tak přibližnou pozici nejsme schopni odhadnout vůbec (koncovky .COM, .NET nebo nové koncovky .WEBCAM, .TRAVEL a mnoho dalších)[3].

#### Geolokace podle seznamu dostupných bezdrátových sítí Wi-Fi

V současné době se s rozmachem chytrých mobilních telefonů často používá určování polohy pomocí měření dostupných bezdrátových sítí Wi-Fi. Určení polohy probíhá na základě informací o okolních bezdrátových sítích, jejich SSID (identifikátor bezdrátové sítě Wi-Fi – Service Set Identifier), MAC (jednoznačný identifikátor síťového

zařízení – Media Access Control) adres a síly signálu. Tato data se porovnají s již známými údaji a zjistí se přibližná poloha zařízení, resp. uživatele.[1].

### 1.2.2 Aktivní metody

Aktivní metody odhadují fyzickou polohu síťového zařízení měřením nejčastěji latence (zpoždění), tedy doby, za kterou se jeden datový segment přenese od zdroje k cíli a zpět – RTT (obousměrné zpoždění – Round Trip Time). Zpoždění je ovlivněno mnoha okolnostmi, mezi které patří aktuální vytížení linky, přenosová rychlost vedení nebo geografická vzdálenost mezi jednotlivými uzly.

# 2 Použité geolokační databáze

Geolokační databáze se dělí z hlediska přístupu k jejich informacím na komerční a volně dostupné. Komerční databáze mají určité licenční podmínky a poplatky za užívání řádově ve stovkách až tisících dolarů za měsíc, ale většinou poskytují více údajů než databáze volně dostupné. Na druhé straně jsou zde volně dostupné geolokační databáze, které nejsou zatížené poplatky, ovšem mají určitá pravidla užívání, např. maximální počet dotazů za den. Údaje ve volně dostupných databázích bývají aktualizovány od různých přispěvovatelů – dobrovolníků, ISP (poskytovatel internetového připojení – Internet Service Provider), různých internetových projektů apod.

## 2.1 Volně dostupné databáze

V této kapitole se budu věnovat jednotlivým volně dostupným geolokačním databázím, které ve své aplikaci implementuji.

### 2.1.1 DB-IP/IPtoCity

Databáze DB-IP obsahuje více než 22 milionů bloků IP adres IPv4 a IPv6[4], což ji řadí mezi jednu z nejobsáhlejších databází. Pro naše účely lze využít verzi IPtoCity, která je k dispozici zdarma. Musíme se však spokojit s faktem, že neobsahuje všechny bloky IP adres, které jsou dostupné v placené verzi (obsahuje přibližně 4 miliony bloků). Poskytované informace nedosahují takové přesnosti jako v případě placených variant[5].

### 2.1.2 HostIP

HostIP je komunitní projekt na geografickou lokalizaci IP adres. Informace o IP adresách a jejich fyzických pozicích, resp. opravy fyzických pozic mohou zadávat všichni uživatelé této služby, protože se jedná o otevřený projekt[6]. Odkud pochází základní data o IP adresách jsem na internetových stránkách tohoto projektu nenalezl, ale vzhledem k faktu, že reklamují možnost získávání přesnějších informací z komerční geolokační databáze MaxMind, tak předpokládám, že určitá část databáze pochází právě odtud[7].

### 2.1.3 ipstack

Ipstack je geolokační databáze, která vyrostla z dříve založeného open-source komunitního projektu freegeoip.net[8]. Služba poskytuje webové API pro získávání

informací o IP adresách. K dispozici jsou informace jako je stát, kraj, město, poštovní směrovací číslo, GPS (globální polohovací systém – Global Positioning System) pozice, měna nebo časová zóna. V bezplatné variantě (s povinnou registrací) je povoleno maximálně 10 000 dotazů za hodinu. Další již placené varianty umožňují vyšší počet dotazů a větší množství získávaných dat[9].

### 2.1.4 MaxMind/GeoLite2City

Geolokační databáze GeoLite2 od společnosti MaxMind je poskytována zdarma, avšak je méně přesná oproti komerční databázi GeoIP2. K databázi GeoLite2 společnost neposkytuje žádnou podporu a vzhledem k licenční politice je povinnost při využívání této databáze uvádět zpětný odkaz na stránky společnosti. Databáze GeoLite2 je poskytována ve dvou verzích – Country a City, použijeme verzi City, protože kromě informací o státu, kde IP adresa leží, obsahuje i další informace o městech a krajích[10].

### 2.1.5 IP2Location/DB5.LITE

Společnost IP2Location poskytuje několik variant svých geolokačních databází v závislosti na tom, jakou přesnost a jaké množství dat potřebujeme. Opět jsou k dispozici jak placené, tak neplacené varianty (verze LITE). V našem případě připadá do úvahy databáze alespoň typu DB5.LITE, protože poskytuje informace o státě, kraji, městě a GPS souřadnicích[11].

## 2.2 Komerční databáze

V následujících podkapitolách budou blíže představeny komerční geolokační databáze. I přes to, že jsou databáze komerční, lze je za určitých podmínek využít (jedná se převážně o velmi nízký limit na počet dotazů za hodinu, resp. za 24 hodin).

### 2.2.1 DB-IP/IPtoLocation

Jak již bylo uvedeno výše, databáze DB-IP obsahuje více než 22 milionů bloků IP adres IPv4 a IPv6. Přes webové rozhraní lze otestovat komerční verzi IPtoLocation této databáze, která oproti volně dostupné verzi obsahuje i informace o GPS souřadnicích a časovém pásmu. Obsahuje také více záznamů oproti verzi IPtoCity[4].

### 2.2.2 MaxMind/GeoIP2City

Společnost MaxMind poskytuje komerční databázi GeoIP2 v několika verzích, pro naše účely je postačující verze City. Komerční databáze GeoIP2 je přesnější a častěji aktualizovaná než volně dostupná databáze GeoLite2[12]. Samotná aplikace pak bude implementovat GeoIP2 Precision Services, což je webové rozhraní pro získávání dat.

### 2.2.3 IP2Location/DB24

Společnost IP2Location založená v roce 2002 poskytuje několik typů svých komerčních geolokačních databází. Pro naše využití by postačovala verze DB5, ovšem společnost na svých webových stránkách poskytuje možnost 50 požadavků za den do své plnohodnotné databáze DB24, a proto v mé aplikaci využiji této možnosti[13].

### 2.2.4 Neustar

Neustar je americká společnost s hlavním sídlem ve městě Sterling ve státě Virginia. Poskytuje řešení ve 5 základních odvětvích - marketing, bezpečnost, řízení rizik, komunikace a správa domén. Pro tato odvětví společnost vyvinula mnoho různých nástrojů a mezi nimi je geolokační databáze, jejíž část je veřejně dostupná přes webové rozhraní[14]. Bohužel ale nebylo možné dohledat bližší informace o jejich databázi - ani počet záznamů, ani množství poskytovaných údajů a ani omezení jejich webového rozhraní.

### 2.2.5 Geobytes

Společnost Geobytes poskytuje geolokaci ve formě Get City Details API, což je volně dostupné rozhraní pro geolokaci IP adresy s limitem 16 384 dotazů za hodinu a bez zabezpečeného připojení SSL (zabezpečený komunikační protokol používaný pro šifrování přenosu informací po síti – Secure Sockets Layer). Pro vyšší počet dotazů, resp. pro využití zabezpečeného SSL přístupu je nutné pořídit odpovídající licenci. Tato databáze oproti ostatním poskytuje např. informaci o počtu obyvatel daného státu nebo používané měně[15].

### 2.2.6 Skyhook

V roce 2003 společnost Skyhook vyvinula vlastní řešení na odhad fyzické polohy zařízení. Řešení je založené na geolokaci pomocí seznamu dostupných Wi-Fi sítí a nespoléhá se na signál GPS ani na mobilní sítě[16]. Jejich geolokační databázi je možné použít v rámci jejich služby Hyperlocal IP, která má dostupné API.

### 2.2.7 ipinfo

Databáze ipinfo poskytuje širokou škálu údajů, mezi které se řadí informace o fyzické lokaci IP adresy, ale také informace o autonomních systémech nebo vlastníkovi adresy. Do 1 000 dotazů za den je geolokační služba zdarma, při vyšší zátěži je nutno zvolit odpovídající předplatné[17].

### 2.2.8 EurekAPI

Tato databáze je produktem společnosti IP-GeoLoc a poskytuje velmi mnoho údajů včetně informací o zeměpisné šířce a délce, poskytovateli internetového připojení apod. Databáze umožňuje získat 30 denní přístup zdarma pro vyzkoušení geolokačních schopností této databáze, když o sobě tvrdí, že pokrývají 99 % bloků všech IP adres. Pro všechny typy licencí jsou ale aplikována pravidla na maximální počet 300 dotazů za minutu, resp. 10 000 dotazů za den[18].

### 2.2.9 ipdata

Jedná se o geolokační databázi agregující data z více zdrojů s volně dostupnými daty GeoLite2 od společnosti MaxMind[17]. Provozovatel této databáze, resp. tohoto API mi sdělil, že mají koncové body v 10 datacentrech – 4 ve Spojených státech amerických, 2 v Evropě (Londýn - Velká Británie, Frankfurt - Německo) a po jednom bodě v Soulu (Jižní Korea), Sydney (Austrálie) a Bombaji (Indie) chlubící se méně než 50 ms zpožděními.

# 3 Druhy kybernetických hrozeb

V dnešní době dochází stále častěji ke kybernetickým útokům v síti Internet. V této kapitole budou popsány vybrané typy útoků, které budou později analyzovány. Útoky se zaměřují na software, hardware nebo síť jako takovou. Cílem útoku je obvykle napadený objekt vyřadit z provozu, zhoršit jeho dostupnost, pozměnit uložená nebo přenášená data, případně jej naopak zapojit do dalších útoků[20].

## 3.1 Nevyžádaná pošta (spam)

Pod pojmem nevyžádaná pošta či spam se obvykle rozumí nevyžádané e-maily, které většinou obsahují reklamu na viagru, recepty na hubnutí, provize za převod milionů dolarů přes účet uživatele nebo se snaží z uživatele vylákat peníze nebo citlivé údaje platebních karet či přístupové údaje k internetovému bankovnictví. Tyto e-maily nejsou cílené na konkrétního uživatele nebo konkrétní skupinu uživatelů – nevyžádaná pošta chodí všem uživatelům bez rozlišení pohlaví, věku nebo pracovního zařazení. K rozesílání nevyžádané pošty jsou obvykle zneužívány špatně zabezpečené poštovní servery nebo zneužité počítače koncových uživatelů. Jako odesílatel je uváděna neexistující e-mailová adresa [21][22].

Nevyžádaná pošta je pro koncové uživatele obtěžující a znepříjemňuje jim používání elektronické pošty – uživatel je nucen procházet nevyžádané e-maily, rozlišovat, zda se jedná o spam či nikoliv, což stojí čas a v konečném případě i peníze. Nevyžádaná pošta zatěžuje poštovní servery, zaplňuje kapacitu e-mailové schránky, zpomaluje stahování nových zpráv a zvyšuje objem přenesených dat, což je problém především u mobilního připojení k Internetu. Sofistikovaná řešení na automatickou detekci nevyžádané pošty nejsou 100%, což má za následek označení normální pošty jako nevyžádanou a naopak[23].

## 3.2 Nevyžádané příspěvky v diskuzních fórech (forum spam)

Anglický pojem „forum spam“ nebo český ekvivalent diskuzní spam či nevyžádané příspěvky v diskuzních fórech je analogií k e-mailové nevyžádané poště. Čtenáři elektronických magazínů, zpravodajských portálů nebo blogů obvykle mívají možnost komentovat články nebo poskytovat zpětnou vazbu majiteli webu prostřednictvím tzv. knihy návštěv. Dále existuje velké množství diskuzních fór, kde lidé mohou mezi sebou diskutovat. Jako „forum spam“ se pak považují příspěvky, které obsahují irelevantní, obtěžující, nevhodný, reklamní nebo nesmyslný obsah. Tyto příspěvky

mohou vkládat sami uživatelé nebo automatizovaní roboti. Chránit se před těmito aktivitami je možné například ochranou proti robotům, zakázáním vkládání odkazů do příspěvků, moderováním diskuze nebo blokováním spammerů na základě IP adresy pomocí některého volně dostupného seznamu[24].

## 3.3 Škodlivý software (malware)

Škodlivý software má za úkol zajistit útočníkovi přístup do nakaženého zařízení (počítač, chytrý telefon, tablet). Malware je velmi široký pojem. Jako malware se označuje všechen software, který byl vytvořen se škodlivým záměrem. Malware je zkratka z anglických slov malicious (škodlivý) a software (programové vybavení). Mezi škodlivý software patří počítačové viry, červy, trojské koně, spyware, adware, phishing, ransomware a další.

Škodlivý software se šíří přes síť Internet a pomocí nevyžádané pošty. Nejčastěji se využívá neznalosti samotných uživatelů, kdy k nakažení zařízení stačí pouhé „ukliknutí“. Další způsoby šíření spočívají ve spuštění programů a aplikací stažených z neoficiálních zdrojů a podezřelých internetových stránek[25][26].

## 3.4 Vyděračský software (ransomware)

Vyděračský software (anglicky ransomware) je typem škodlivého softwaru, který má za cíl zašifrovat data nakaženého zařízení. Následně ransomware požaduje výpalné výměnou za sdělení hesla nutného k dešifrování dat, což ovšem útočníci nijak negarantují. V některých případech vyděračský software uživateli zobrazuje informaci, kolik času zbývá na zaplacení výpalného, příp. kolik času zbývá do navýšení požadované sumy. Výpalné je obvykle požadováno zaplatit ve virtuálních měnách, např. bitcoin[27][28].

## 3.5 Útoky na redakční systém WordPress

WordPress je open-source redakční systém určený k provozu internetových stránek, blogů a webových aplikací. Může se chlubit více než 60% tržním podílem, který si získal jednoduchostí, použitím běžně dostupných technologií (PHP, MySQL) a širokou škálou doplňků a šablon. Systém má poměrně bohatou historii a prošel velkým vývojem i v oblasti bezpečnosti. O bezpečnost jádra systému se stará přibližně 50 bezpečnostních expertů, nicméně situace ohledně bezpečnosti doplňků je horší. Jelikož je systém používán více než třetinou z 10 milionů největších webových

stránek a dále velkým množstvím malých webů, tak se na tento redakční systém samozřejmě soustředí i útočníci[29][30].

Útočníci využívají chyb typu SQL injection nebo cross-site scripting, což jim umožní pozměnit obsah webových stránek nebo získat nad redakčním systémem částečnou nebo úplnou kontrolu. Tyto weby pak útočníci využívají k šíření škodlivého softwaru, zobrazování reklamy nebo ke zneužití e-mailových adres registrovaných uživatelů k rozesílání nevyžádané pošty[31].

## 3.6 Síť Tor

Síť Tor je uživateli využívána k anonymnímu přístupu k síti Internet. Uživatelům pomáhá skrýt údaje o IP adrese a dalších údajích, díky čemuž je obtížné vysledovat činnost uživatele na Internetu. Síť Tor se skládá z mnoha uzlů umístěných po celém světě. Komunikace prostřednictvím této sítě probíhá od klienta přes několik uzlů až k výstupnímu uzlu, který vytvoří spojení k serveru umístěnému v Internetu. Cílový server ví pouze o spojení s výstupním uzlem. Údaje mezi uzly uvnitř sítě Tor jsou šifrované a neznají cílový server[32].

Komunikace po síti Tor chrání svobodu a soukromí uživatelů. Přednosti této sítě lze však zneužít, resp. využít k nelegálním činnostem. Z těchto důvodů byl popis sítě Tor zařazen do této kapitoly a následně bude analyzována geografická lokalizace tzv. výstupních uzlů, které lze považovat za slabá místa celé komunikace (místo kde přechází šifrovaná a anonymizovaná komunikace do veřejné sítě Internet)[33][34].

# 4 Vývoj aplikace `ip2geotools`

Zadáním práce je vytvořit aplikaci, která bude provádět odhad geografické polohy pro zadanou IP adresu. Moji aplikaci jsem nazval `ip2geotools`. Požadavkem je, aby aplikace byla naprogramována v jazyce Python a spustitelná pod operačním systémem na bázi GNU/Linux.

Ve spolupráci s mým vedoucím práce jsme stanovili stěžejní body aplikace. Aplikace:

- musí být rozšiřitelná o další geolokační databáze,
- musí obsahovat jednoduchý způsob získání geolokačních informací z vybrané databáze o zadané IP adrese,
- musí být použitelná bez nutnosti instalovat velké množství dalšího softwaru, který nesouvisí s jazykem Python,
- měla by být využitelná i jinými subjekty než mojí fakultou, resp. mojí univerzitou.

## 4.1 Návrh aplikace

Nejdůležitějším krokem při vytváření jakékoliv aplikace nebo složitějšího systému je jeho pečlivý a promyšlený návrh. Jelikož požadavky předurčují aplikaci pro její další rozšiřování a vylepšování, je nutné zvolit odpovídající způsob provedení, a proto při vytváření této aplikace budou v maximální možné míře použity principy OOP (objektově orientované programování – Object-Oriented Programming).

Aplikace musí být rozdělena na menší části, které budou znovupoužitelné i mimo tuto aplikaci. Pro tuto aplikaci, resp. její dílčí části, se přímo vybízí aplikovat unixovou filosofii KISS (Zachovej to jednoduché, hlupáku! – Keep It Simple, Stupid!), kterou shrnul Eric Raymond ve své knize The Art of Unix Programming[35] do těchto bodů:

- **Pravidlo modularity**: Pište jednoduché části, které budou spojeny bezchybným rozhraním.
- **Pravidlo srozumitelnosti**: Srozumitelnost je lepší než šikovnost.
- **Pravidlo kompozice**: Navrhujte programy tak, aby mohly být napojeny na jiné programy.
- Pravidlo oddělování: Oddělujte postupy od mechanismů; oddělujte rozhraní od nástrojů.
- **Pravidlo jednoduchosti**: Programujte jednoduše; složitě jen tam, kde je to nezbytné.
- Pravidlo úspornosti: Pište rozsáhlé programy jen je-li zřejmé, že to jiným způsobem nejde.

- **Pravidlo přehlednosti**: Programujte přehledně, aby byly audity kódu a odstraňování chyb jednodušší.
- Pravidlo robustnosti: Robustnost je potomek přehlednosti a jednoduchosti.
- Pravidlo reprezentace: Převeďte znalosti do kódu, aby mohl být program hloupý a robustní.
- Pravidlo nejméně překvapivého: Při návrhu rozhraní vždy dělejte tu nejméně překvapivou věc.
- Pravidlo ticha: Nemá-li program co nového říct, neměl by říkat nic.
- **Pravidlo opravy**: Musíte-li selhat, selžete hlasitě a co nejdříve.
- **Pravidlo úspornosti**: Programátorův čas je drahý, šetřte ho použitím strojového času.
- Pravidlo rozmnožování: Vyhněte se ručnímu psaní kódu; kdykoliv to jde, pište programy, které píšou další programy.
- Pravidlo optimalizace: Vytvořte správně fungující program, než ho začnete optimalizovat.
- **Pravidlo pestrosti**: Nevěřte pouze v jedno správné řešení.
- **Pravidlo rozšiřitelnosti**: Plánujte do budoucnosti, protože ta přijde dříve, než si myslíte.

Tučně zvýrazněné body jsou ty, které se budu snažit v mém návrhu aplikovat. Tudíž aplikace bude:

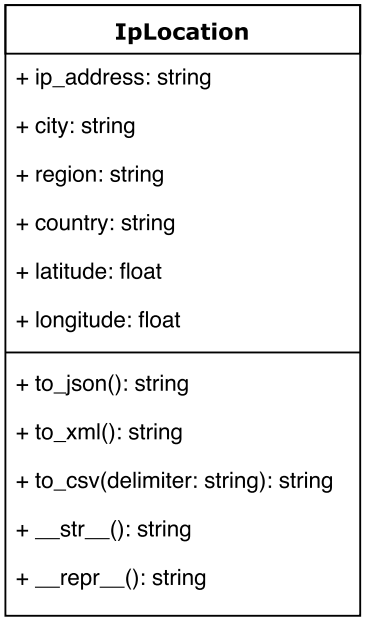
- rozdělena na menší části, pro jednotlivé geolokační databáze bude vytvořeno rozhraní (pravidlo modularity, rozšiřitelnosti a pestrosti),
- postavena na obecně rozšířených postupech a bude dodržovat zavedené zvyklosti (pravidlo srozumitelnosti, jednoduchosti a přehlednosti),
- moct být spuštěna z příkazového řádku a bude vracet data čistě v textovém formátu, což umožní napojení na další programy (pravidlo kompozice),
- využívat bloků try-catch a bude obsahovat sadu vlastních typů výjimek (pravidlo opravy)
- pro svoji efektivní činnost využívat již naprogramované moduly (pravidlo úspornosti).

## 4.2 Formát a struktura dat

Modul `ip2geotools.models` je určen k uchovávání tříd – modelů využívaných v rámci balíčku `ip2geotools`, ale samozřejmě použitelných i mimo něj. Model je třída určená k uchovávání informací, obsahuje vlastnosti a zpracovává data předem daným způsobem.

### `IpLocation`

Model `IpLocation` je momentálně jediným modelem modulu `ip2geotools.models`. Tento model slouží k reprezentaci polohy IP adresy. V případě potřeby je možné ho jednoduše rozšířit o další vlastnosti. Model je využíván v rámci balíčku `ip2geotools` všude tam, kde je potřeba zajistit jednotnou návratovou hodnotu obsahující geografickou lokalizaci IP adresy.




Obr. 4.1: Struktura modelu `IpLocation`

Ze struktury modelu na obr. 4.1 je patrné, že model obsahuje následující vlastnosti:

- `ip_address` – IP adresa,
- `city` – město, kde se nachází IP adresa,
- `region` – kraj, kde se nachází IP adresa,
- `country` – stát, kde se nachází IP adresa (dvoupísmenný kód dle ISO 3166-1 alpha-2),
- `latitude` – zeměpisná šířka, kde se nachází IP adresa,
- `longitude` – zeměpisná délka, kde se nachází IP adresa,

a že model dále obsahuje také tyto metody:

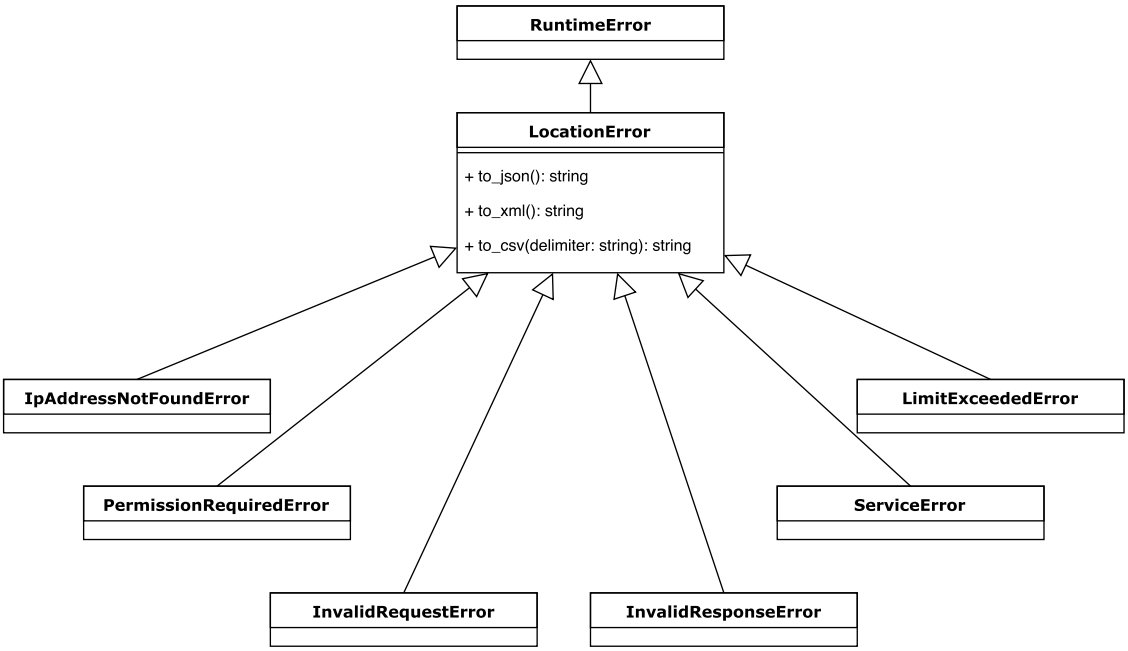
- `to_json` – metoda vrací data modelu ve formátu JSON (JavaScriptový objektový zápis – JavaScript Object Notation),
- `to_xml` – metoda vrací data modelu ve formátu XML (rozšiřitelný značkovací jazyk – Extensible Markup Language); jako kořenový element je zvolen název

`ip_location`,

- `to_csv` – metoda vrací data modelu ve formátu CSV (hodnoty oddělené čárkami – Comma-Separated Values); oddělovač se uvádí jako parametr této metody,
- `__str__` – metoda, kterou obsahuje každá třída jazyka Python; tato metoda vrací data modelu jako naformátovaný text, kdy každá informace je uvedena na novém řádku,
- `__repr__` – metoda, kterou obsahuje každá třída jazyka Python, tato metoda vrací krátký řetězec obsahující informaci, z jakého balíčku a modulu tento model pochází a o jaké IP adrese nese data.

## 4.3 Definované typy použitých výjimek

Modul `ip2geotools.errors` uchovává třídy – výjimky, které jsou určené v rámci balíčku `ip2geotools` k detailnímu rozlišení chyb, které mohou během geolokace nastat. Výjimky jsou datové struktury nesoucí informace o chybovém stavu. Zpřehlední a ve výsledku zjednoduší zdrojový kód a zároveň podpoří modulární návrh celé aplikace.




Obr. 4.2: Diagram výjimek

Struktura výjimek je zachycena na obr. 4.2, kde můžeme vidět, že jsem vytvořil obecnou výjimku `LocationError` děděním z třídy `RuntimeError`. Tento způsob zavedení vlastních výjimek považuji za logický a v praxi běžně používaný – zajistím

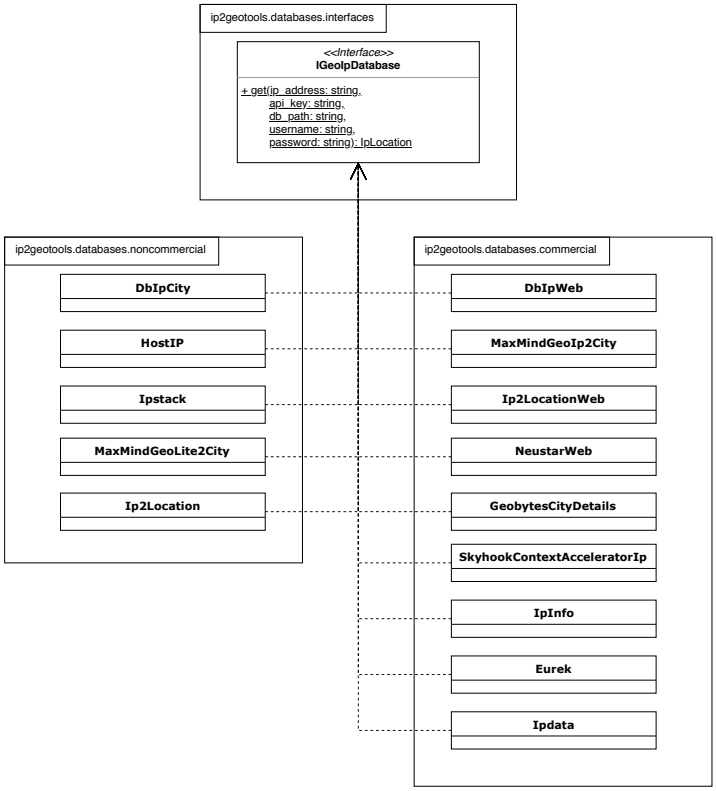
si tak, že další typy výjimek budou mít společného rodiče a dohromady budou tvořit logickou hiearchii (tzn. že všechny další typy výjimek ponesou zároveň informaci o tom, že se jedná o problém s geolokací).

Během geolokace mohou nastat jisté chyby a problémy, a proto jsem děděním z obecné geolokační výjimky `LocationError` zavedl následujících 6 výjimek:

- `IpAddressNotFoundError` – IP adresa nebyla nalezena,
- `PermissionRequiredError` – nastal problém s autentizací nebo autorizací požadavku; zkontrolujte oprávnění pro přístup do geolokační databáze,
- `InvalidRequestError` – neplatný požadavek,
- `InvalidResponseError` – neplatná odpověď,
- `ServiceError` – odpověď geolokační databáze není platná (databáze není dostupná, apod.),
- `LimitExceededError` – bylo dosaženo limitů stanovených danou geolokační databází.

## 4.4 Začleněné databáze

Geolokační databáze podporují různé způsoby získávání dat, a to většinou přes webové rozhraní nebo pomocí různých knihoven. V této části práce popíšu, jaké způsoby napojení na geolokační databáze jsem zvolil a jak jsem je implementoval. Tento modul `ip2geotools.databases` je rozdělen na 3 dílčí moduly, a to `interfaces`, `noncommercial` a `commercial`, které obsahují společné rozhraní, resp. třídy pro práci s jednotlivými databázemi implementující společné rozhraní, což lze vidět na obr. 4.3.




Obr. 4.3: Diagram tříd pro získávání dat z geolokačních databází

### 4.4.1 Navržené vlastní rozhraní

Modul `ip2geotools.databases.interfaces` je určen pro uchování jediného rozhraní `IGeoIpDatabase` (viz 4.1), které definuje pro všechny implementace geolokačních databází jednotný způsob získávání dat. Obecně lze říci, že v jazyce Python nejsou nutná rozhraní, protože jazyk umožňuje třídu dědit z více jiných tříd.

Z jiných programovacích jazyků jsem zvyklý definovat rozhraní tam, kde se to přímo nabízí, a to je přesně tento případ – k tomu musím využít způsob nazývaný „Abstract Base Class“, zkráceně ABC[36]. Rozhraní pak definuje jedinou metodu `get`, která musí obsahovat několik parametrů – IP adresu, klíč k API (pokud je nutný), cestu k databázovému souboru (pokud se jedná o lokálně staženou databázi) a jméno s heslem (pokud je geolokační databáze přístupná po přihlášení).

```
1 from abc import ABCMeta, abstractmethod
2
3 class IGeoIpDatabase:
4     __metaclass__ = ABCMeta
5
6     @staticmethod
7     @abstractmethod
8     def get(ip_address, api_key, db_path, username, password):
9         raise NotImplementedError
```

Výpis 4.1: Navržené rozhraní `IGeoIpDatabase`

### 4.4.2 Volně dostupné databáze

#### DB-IP/IPtoCity

API této služby je postaveno na pricipech REST (Representational State Transfer). Informace z API se získávají jednoduchými HTTP (Hypertext Transfer Protocol) požadavky (samotný návrh REST umožňuje požadavky GET, POST, PUT a DELETE), toto API umožňuje pouze GET požadavky.

Z API lze získávat informace následovně:

Příklad požadavku:

```
1 http://api.db-ip.com/addrinfo?api_key=API_KLIC&addr
    =147.229.2.90
```

Odpověď:

```
1 {
2   "address": "147.229.2.90",
3   "country": "CZ",
4   "stateprov": "South Moravian",
5   "city": "Brno (Brno střed)"
6 }
```

API využívající principů návrhu REST pracují především ve formátu JSON. S formátem JSON jsem obeznámen z prací na mnoha jiných projektech a osobně mi vyhovuje, protože informace jsem schopen z odpovědi ověřit letmým pohledem. Práce s JSON objekty je v mnoha jazycích poměrně jednoduchá a pohodlná a jinak tomu není ani v programovacím jazyce Python.

Na výpisu 4.2 je ukázka třídy `DbIpCity` implementující rozhraní `IGeoIpDatabase`, která ověřuje IP adresu zadanou v parametru metody `get` vůči databázi DB-IP-/IPtoCity. Dalším povinným parametrem této metody je klíč k API. Nejdříve se provede HTTP GET požadavek na zadanou URL (jednotná adresa zdroje – Uniform Resource Locator) adresu včetně patřičného klíče a zadané IP adresy. V případě, že API vrátí regulérní odpověď, tak HTTP hlavička obsahuje stav 200 OK, v ostatních případech je vyhozena výjimka `ServiceError`. Dále se obsah odpovědi převede na JSON objekty. V případě, že by se převod nezdařil, tak se vyhodí výjimka `InvalidResponseError`. Obsah odpovědi se zkontroluje na chybové zprávy

týkající se nenalezení IP adresy nebo neplatnosti API klíče a podle toho se vyhodí patřičná výjimka.

Jelikož databáze v této variantě neposkytuje GPS pozice, implementoval jsem alespoň základní získání těchto souřadnic pomocí Open Street Map Nominatim, kde se vyhledají souřadnice řetězce „město, kraj stát“, příp. „město, stát“. Metoda vrací objekt třídy `IpLocation` naplněný získanými daty.

```
class DbIpCity(IGeoIpDatabase):
    """
    Class for accessing geolocation data provided by https://db-ip.com/api/.
    """

    @staticmethod
    def get(ip_address, api_key='free', db_path=None, username=None, password=None)
        :
        # process request
        try:
            request = requests.get('http://api.db-ip.com/v2/'
                                   + quote(api_key)
                                   + '/' + quote(ip_address),
                                   timeout=62)
        except:
            raise ServiceError()

        # check for HTTP errors
        if request.status_code != 200:
            raise ServiceError()

        # parse content
        try:
            content = request.content.decode('utf-8')
            content = json.loads(content)
        except:
            raise InvalidResponseError()

        # check for errors
        if content.get('error'):
            if content['error'] == 'invalid address':
                raise IpAddressNotFoundError(ip_address)
            elif content['error'] == 'invalid API key':
                raise PermissionRequiredError()
            else:
                raise InvalidRequestError()

        # prepare return value
        ip_location = IpLocation(ip_address)

        # format data
        ip_location.country = content['countryCode']
        ip_location.region = content['stateProv']
        ip_location.city = content['city']

        # get lat/lon from OSM
        osm = geocoder.osm(ip_location.city + ', '
                           + ip_location.region + ' '
                           + ip_location.country,
                           timeout=62)
```

```
50
51          if osm.ok:
52              osm = osm.json
53              ip_location.latitude = float(osm['lat'])
54              ip_location.longitude = float(osm['lng'])
55          else:
56              osm = geocoder.osm(ip_location.city + ', ' + ip_location.country,
                    timeout=62)
57
58              if osm.ok:
59                  osm = osm.json
60                  ip_location.latitude = float(osm['lat'])
61                  ip_location.longitude = float(osm['lng'])
62              else:
63                  ip_location.latitude = None
64                  ip_location.longitude = None
65
66          return ip_location
```

Výpis 4.2: Ukázka implementace třídy `DbIpCity`

### HostIP

K API této služby se přistupuje běžnými HTTP GET požadavky. API umožňuje vracet následující data:

(a) dvoupísmenný kód země

Příklad požadavku:

```
1 http://api.hostip.info/country.php?ip=147.229.2.90
```

Odpověď:

```
1 CZ
```

(b) dvoupísmenný kód země, název země, název města, GPS souřadnice a zadanou IP adresu ve formátu plaintext (obyčejný text)

Příklad požadavku:

```
1 http://api.hostip.info/get_html.php?ip=147.229.2.90&
    position=true
```

Odpověď:

```
1 Country: CZECH REPUBLIC (CZ)
2 City: Brno
3
4 Latitude: 49.2
5 Longitude: 16.6333
6 IP: 147.229.2.90
```

(c) dvoupísmenný kód země, název země, název města, GPS souřadnice a zadanou IP adresu ve formátu JSON

Příklad požadavku:

```
http://api.hostip.info/get_json.php?ip=147.229.2.90&
    position=true
```

Odpověď:

```
{
  "country_name": "CZECH REPUBLIC",
  "country_code": "CZ",
  "city": "Brno",
  "ip": "147.229.2.90",
  "lat": "49.2",
  "lng": "16.6333"
}
```

(d) dvoupísmenný kód země, název země, název města, GPS souřadnice a zadanou IP adresu ve formátu XML

Příklad požadavku:

```
http://api.hostip.info/?ip=147.229.2.90
```

Odpověď:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<HostipLookupResultSet version="1.0.1"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="http://www.hostip.info/
      api/hostip-1.0.1.xsd">
  <gml:description>This is the Hostip Lookup Service</
      gml:description>
  <gml:name>hostip</gml:name>
  <gml:boundedBy>
    <gml:Null>inapplicable</gml:Null>
  </gml:boundedBy>
  <gml:featureMember>
    <Hostip>
      <ip>147.229.2.90</ip>
      <gml:name>Brno</gml:name>
      <countryName>CZECH REPUBLIC</countryName>
      <countryAbbrev>CZ</countryAbbrev>
```

```
16          <!-- Co-ordinates are available as lng,lat -->
17          <ipLocation>
18            <gml:pointProperty>
19              <gml:Point srsName="http://www.opengis.net/gml/srs
                    /epsg.xml#4326">
20                <gml:coordinates>16.6333,49.2</gml:coordinates>
21              </gml:Point>
22            </gml:pointProperty>
23          </ipLocation>
24        </Hostip>
25    </gml:featureMember>
26 </HostipLookupResultSet>
```

Pro implementaci této geolokační databáze jsem zvolil variantu C, tedy odpovědi API ve formátu JSON. Výpis 4.3 zobrazuje třídu `HostIp` s metodou `get`, která ověřuje IP adresu zadanou v parametru vůči databázi HostIP. Metoda nejdříve provede HTTP GET požadavek na patřičnou URL adresu, kde se specifikuje zadaná IP adresa. Odpověď je zkontrolována na HTTP hlavičku 200 OK, protože jiná hlavička by znamenala, že server vrátil neplatnou odpověď – v takovém případě je vyhozena výjimka podle typu zjištěné chyby. Následně se JSON řetězec z odpovědi převede na objekt. Abych zajistil jednotný formát výstupních dat z jednotlivých databází, musí se data obsažená v JSON zkontrolovat a případně upravit, protože může nastat situace, kdy zadaná IP adresa neexistuje (API ji vrátí jako privátní adresu), případně databáze neobsahuje určité údaje o konkrétní IP adrese (nezná stát nebo město nebo GPS souřadnice). Následně metoda vrací výsledek jako objekt typu `IpLocation` naplněný patřičnými daty.

```
1  class HostIp(IGeoIpDatabase):
2      """
3      Class for accessing geolocation data provided by http://hostip.info/.
4      """
5
6      @staticmethod
7      def get(ip_address, api_key=None, db_path=None, username=None, password=None):
8          # process request
9          try:
10             request = requests.get('http://api.hostip.info/get_json.php?position=
                   true&ip='
11                                    + quote(ip_address),
12                                    timeout=62)
13         except:
14             raise ServiceError()
15
16         # check for HTTP errors
17         if request.status_code != 200:
18             if request.status_code == 404:
19                 raise IpAddressNotFoundError(ip_address)
```

```
20              elif request.status_code == 500:
21                  raise InvalidRequestError()
22              else:
23                  raise ServiceError()
24
25          # parse content
26          try:
27              content = request.content.decode('utf-8')
28              content = json.loads(content)
29          except:
30              raise InvalidResponseError()
31
32          # prepare return value
33          ip_location = IpLocation(ip_address)
34
35          # format data
36          if content['country_code'] == 'XX':
37              ip_location.country = None
38          else:
39              ip_location.country = content['country_code']
40
41          ip_location.region = None
42
43          if content['city'] == '(Unknown City?)' \
44             or content['city'] == '(Unknown city)' \
45             or content['city'] == '(Private Address)':
46              ip_location.city = None
47          else:
48              ip_location.city = content['city']
49
50          if content.get('lat') and content.get('lng'):
51              ip_location.latitude = float(content['lat'])
52              ip_location.longitude = float(content['lng'])
53          else:
54              ip_location.latitude = None
55              ip_location.longitude = None
56
57          return ip_location
```

Výpis 4.3: Ukázka implementace třídy `HostIp`

### ipstack

K API této služby se přistupuje obdobně jako v předešlých případech HTTP GET požadavky. Z API lze získávat veškerá data v několika formátech následovně:

(a) XML

Příklad požadavku:

```
1 http://api.ipstack.com/147.229.2.90?access_key=API_KLIC&
      output=xml
```

Odpověď:

```
<?xml version="1.0"?>
<result>
  <ip>147.229.2.90</ip>
  <type>ipv4</type>
  <continent_code>EU</continent_code>
  <continent_name>Europe</continent_name>
  <country_code>CZ</country_code>
  <country_name>Czechia</country_name>
  <region_code>64</region_code>
  <region_name>South Moravian</region_name>
  <city>Brno</city>
  <zip>614 00</zip>
  <latitude>49.2333</latitude>
  <longitude>16.65</longitude>
  <location>
    <geoname_id>3078610</geoname_id>
    <capital>Prague</capital>
    <languages>
      <code>cs</code>
      <name>Czech</name>
      <native>&amp;#x10C;esky</native>
    </languages>
    <languages>
      <code>sk</code>
      <name>Slovak</name>
      <native>Sloven&amp;#x10D;ina</native>
    </languages>
    <country_flag>http://assets.ipstack.com/flags/cz.svg</
        country_flag>
    <country_flag_emoji>&amp;#x1F1E8;&amp;#x1F1FF;</
        country_flag_emoji>
    <country_flag_emoji_unicode>U+1F1E8 U+1F1FF</
        country_flag_emoji_unicode>
    <calling_code>420</calling_code>
    <is_eu>1</is_eu>
  </location>
</result>
```

(b) JSON

Příklad požadavku:

```
http://api.ipstack.com/147.229.2.90?access_key=API_KLIC&
    output=json
```

Odpověď:

```
{
  "ip": "147.229.2.90",
  "type": "ipv4",
  "continent_code": "EU",
  "continent_name": "Europe",
  "country_code": "CZ",
  "country_name": "Czechia",
  "region_code": "64",
  "region_name": "South Moravian",
  "city": "Brno",
  "zip": "614 00",
  "latitude": 49.2333,
  "longitude": 16.65,
  "location": {
    "geoname_id": 3078610,
    "capital": "Prague",
    "languages": [
      {
        "code": "cs",
        "name": "Czech",
        "native": "Česky"
      },
      {
        "code": "sk",
        "name": "Slovak",
        "native": "Slovenčina"
      }
    ],
    "country_flag": "http://assets.ipstack.com/flags/cz.svg"
      ,
    "country_flag_emoji": "xx",
    "country_flag_emoji_unicode": "U+1F1E8 U+1F1FF",
    "calling_code": "420",
    "is_eu": true
```

```
34    }
35 }
```

(c) JSONP (JSON with Padding)

Příklad požadavku:

```
1 http://api.ipstack.com/147.229.2.90?access_key=API_KLIC&
      callback=mojeFunkce
```

Odpověď:

```
1  mojeFunkce({
2    "ip": "147.229.2.90",
3    "type": "ipv4",
4    "continent_code": "EU",
5    "continent_name": "Europe",
6    "country_code": "CZ",
7    "country_name": "Czechia",
8    "region_code": "64",
9    "region_name": "South Moravian",
10   "city": "Brno",
11   "zip": "614 00",
12   "latitude": 49.2333,
13   "longitude": 16.65,
14   "location": {
15     "geoname_id": 3078610,
16     "capital": "Prague",
17     "languages": [
18       {
19         "code": "cs",
20         "name": "Czech",
21         "native": "Česky"
22       },
23       {
24         "code": "sk",
25         "name": "Slovak",
26         "native": "Slovenčina"
27       }
28     ],
29     "country_flag": "http://assets.ipstack.com/flags/cz.svg"
         ,
30     "country_flag_emoji": "xx",
31     "country_flag_emoji_unicode": "U+1F1E8 U+1F1FF",
```

```
32      "calling_code": "420",
33      "is_eu": true
34    }
35  })
```

Použil jsem opět formát odpovědi ve formátu JSON. Metoda `get` na získávání informací o IP adrese v třídě `Ipstack` (viz 4.4) je velmi podobná třídám popsaným výše. Metoda opět zašle HTTP GET požadavek na vybranou URL adresu. Vyčká se na vypršení požadavku – celkem až 62 vteřin. Zkontroluje se odpověď na stavový kód 200 OK, protože jiný kód by znamenal problém se serverem. Rozhraní bohužel používá pro sdělování chybových stavů poměrně nešťastné řešení, a to odpověď s HTTP stavovým kódem 200 OK, kde odpověď obsahuje informaci o našem požadavku, resp. zda došlo k nějaké chybě, což jsem převedl na vyhazování odpovídajících výjimek. Obsah odpovědi se převede na JSON objekt a provede se unifikace chybějících informací v odpovědi. Metoda opět vrací data ve formě objektu třídy `IpLocation`.

```
1  class Ipstack(IGeoIpDatabase):
2      """
3      Class for accessing geolocation data provided by http://ipstack.com/.
4
5      """
6
7      @staticmethod
8      def get(ip_address, api_key=None, db_path=None, username=None, password=None):
9          # process request
10         try:
11             request = requests.get('http://api.ipstack.com/' + quote(ip_address)
12                                    + '?access_key=' + quote(api_key),
13                                    timeout=62)
14         except:
15             raise ServiceError()
16
17         # check for HTTP errors
18         if request.status_code != 200:
19             raise ServiceError()
20
21         # parse content
22         try:
23             content = request.content.decode('utf-8')
24             content = json.loads(content)
25         except:
26             raise InvalidResponseError()
27
28         # check for errors
29         if content.get('error'):
30             if content['error']['code'] == 101 \
31                 or content['error']['code'] == 102 \
32                 or content['error']['code'] == 105:
33                 raise PermissionRequiredError()
34             elif content['error']['code'] == 104:
35                 raise LimitExceededError()
```

```
36              else:
37                  raise InvalidRequestError()
38
39          # prepare return value
40          ip_location = IpLocation(ip_address)
41
42          # format data
43          if content['country_code'] == '':
44              ip_location.country = None
45          else:
46              ip_location.country = content['country_code']
47
48          if content['region_name'] == '':
49              ip_location.region = None
50          else:
51              ip_location.region = content['region_name']
52
53          if content['city'] == '':
54              ip_location.city = None
55          else:
56              ip_location.city = content['city']
57
58          if content['latitude'] != '-' and content['longitude'] != '-':
59              ip_location.latitude = float(content['latitude'])
60              ip_location.longitude = float(content['longitude'])
61          else:
62              ip_location.latitude = None
63              ip_location.longitude = None
64
65          return ip_location
```

Výpis 4.4: Ukázka implementace třídy `Ipstack`

### MaxMind/GeoLite2City

Společnost MaxMind má velmi slušně zpracované API k poskytování informací. V případě placených variant je možné využívat webového rozhraní, v případě neplacené varianty lze využívat čtení z lokálně staženého databázového souboru ve formátech CSV nebo MMDB (MaxMind binární formát). K datům se přistupuje nepřímo, a proto se využívají dodávané knihovny, které jsou dostupné pro nepřeberné množství programovacích jazyků (Python, PHP, .NET C#, C, JAVA nebo Perl), a dále také knihovny třetích stran pro další jazyky (Lua, Ruby, Node.js a další). Uvádět tedy příklady volání každého API nemá vzhledem k počtu rozmanitých verzí smysl. Níže se podíváme, jak jednoduše lze získat informace o IP adrese ze City databáze v Pythonu:

Příklad požadavku:

```
1 >>> import geoip2.database
2 >>> reader = geoip2.database.Reader('GeoLite2-City.mmdb')
```

```
>>> response = reader.city('147.229.2.90')
>>> print response
```

Odpověď: (dekódované unicode znaky)

```
geoip2.models.City({u'city': {u'geoname_id': 3078610, u'names':
    {u'ru': u'\u0411\u0440\u043d\u043e', u'fr': u'Brno', u'en': u
    'Brno', u'de': u'Br\xfcnn', u'pt-BR': u'Brno', u'ja': u'\
    u30d6\u30eb\u30ce', u'es': u'Brno'}}, 'traits': {'ip_address'
    : '147.229.2.90'}, u'country': {u'geoname_id': 3077311, u'
    iso_code': u'CZ', u'names': {u'ru': u'\u0427\u0435\u0448\
    u0441\u043a\u0430\u044f \u0420\u0435\u0441\u043f\u0443\u0431\
    u043b\u0438\u043a\u0430', u'fr': u'R\xe9publique tch\xe8que',
     u'en': u'Czech Republic', u'de': u'Tschechien', u'zh-CN': u'
    \u6377\u514b\u5171\u548c\u56fd', u'pt-BR': u'Rep\xfablica
    Checa', u'ja': u'\u30c1\u30a7\u30b3\u5171\u548c\u56fd', u'es'
    : u'Rep\xfablica Checa'}}, u'registered_country': {u'
    geoname_id': 3077311, u'iso_code': u'CZ', u'names': {u'ru': u
    '\u0427\u0435\u0448\u0441\u043a\u0430\u044f \u0420\u0435\
    u0441\u043f\u0443\u0431\u043b\u0438\u043a\u0430', u'fr': u'R\
    xe9publique tch\xe8que', u'en': u'Czech Republic', u'de': u'
    Tschechien', u'zh-CN': u'\u6377\u514b\u5171\u548c\u56fd', u'
    pt-BR': u'Rep\xfablica Checa', u'ja': u'\u30c1\u30a7\u30b3\
    u5171\u548c\u56fd', u'es': u'Rep\xfablica Checa'}}, u'
    subdivisions': [{u'geoname_id': 3339536, u'iso_code': u'JM',
    u'names': {u'ru': u'\u042e\u0436\u043d\u043e\u043c\u043e\
    u0440\u0430\u0432\u0441\u043a\u0438\u0439 \u043a\u0440\u0430\
    u0439', u'fr': u'Moravie-du-Sud', u'en': u'South Moravian', u
    'de': u'S\xfcdm\xe4hrische Region', u'pt-BR': u'Mor\xe1via do
     Sul', u'ja': u'\u5357\u30e2\u30e9\u30f4\u30a3\u30a2\u5dde',
    u'es': u'Regi\xf3n de Moravia Meridional'}}, {u'geoname_id':
    3078609, u'iso_code': u'622', u'names': {u'en': u'Mesto Brno'
    }}], u'location': {u'latitude': 49.2, u'time_zone': u'Europe/
    Prague', u'longitude': 16.6333}, u'postal': {u'code': u'614
    00'}, u'continent': {u'geoname_id': 6255148, u'code': u'EU',
    u'names': {u'ru': u'\u0415\u0432\u0440\u043e\u043f\u0430', u'
    fr': u'Europe', u'en': u'Europe', u'de': u'Europa', u'zh-CN':
     u'\u6b27\u6d32', u'pt-BR': u'Europa', u'ja': u'\u30e8\u30fc\
    u30ed\u30c3\u30d1', u'es': u'Europa'}}}, ['en'])
```

Třída `MaxMindGeoLite2City` implementující rozhraní `IGeoIpDatabase` pro získávání geografických informací o IP adrese z databáze MaxMind/GeoLite2 je odlišná

od předchozích, protože využívá knihovny `geoip2`, což můžeme vidět na výpisu 4.5. Metoda `get` má kromě povinného parametru specifikujícího IP adresu také druhý parametr určující cestu k MMDB souboru. Metoda načte databázový soubor a vyhledají se informace o konkrétní IP adrese. Jelikož se o každé IP adrese podaří získat různé množství informací, musí se opět provést unifikace výstupních dat a naplnění objektu typu `IpLocation`. V případě, že by během nějaké činnosti při práci s knihovnou geoip2 došlo k nějaké výjimce, je tato výjimka odchycena a opětovně je vyhozena odpovídající výjimka z modulu `ip2geotools.errors`.

```
1  class MaxMindGeoLite2City(IGeoIpDatabase):
2      """
3      Class for accessing geolocation data provided by GeoLite2 database
4      created by MaxMind, available from https://www.maxmind.com/.
5      Downloadable from https://dev.maxmind.com/geoip/geoip2/geolite2/.
6      """
7
8      @staticmethod
9      def get(ip_address, api_key=None, db_path=None, username=None, password=None):
10         # process request
11         try:
12             request = geoip2.database.Reader(db_path)
13         except:
14             raise ServiceError()
15
16         # content
17         try:
18             res = request.city(ip_address)
19         except TypeError:
20             raise InvalidRequestError()
21         except geoip2.errors.AddressNotFoundError:
22             raise IpAddressNotFoundError(ip_address)
23
24         # prepare return value
25         ip_location = IpLocation(ip_address)
26
27         # format data
28         if res.country:
29             ip_location.country = res.country.iso_code
30         else:
31             ip_location.country = None
32
33         if res.subdivisions:
34             ip_location.region = res.subdivisions[0].names['en']
35         else:
36             ip_location.region = None
37
38         if res.city.names:
39             ip_location.city = res.city.names['en']
40         else:
41             ip_location.city = None
42
43         if res.location:
44             ip_location.latitude = float(res.location.latitude)
45             ip_location.longitude = float(res.location.longitude)
46         else:
47             ip_location.latitude = None
```

```
48              ip_location.longitude = None
49
50          return ip_location
```

Výpis 4.5: Ukázka implementace třídy `MaxMindGeoLite2City`

### IP2Location/DB5.LITE

Společnost IP2Location má obdobně jako MaxMind velmi slušně zpracované API k poskytování informací. K datům v databázových souborech se jednoduše přistupuje za pomoci knihoven pro konkrétní programovací jazyky (Perl, Python, Pascal, PHP, Ruby a další), které jsou zdarma k dispozici, nebo pomocí knihoven placených (.NET, C#, VB.NET nebo JAVA). Dále je také možnost využít programů pro Windows nebo Linux, resp. multiplatformního Perl skriptu. Uvedu příklad, jak získat informace o IP adrese pomocí `IP2Location` modulu pro Python.

Příklad požadavku:

```
1 >>> import IP2Location
2 >>> ip2loc = IP2Location.IP2Location()
3 >>> ip2loc.open('IP-COUNTRY-REGION-CITY-LATITUDE-LONGITUDE-
    SAMPLE.BIN')
4 >>> print(ip2loc.get_all('147.229.2.90'))
```

Odpověď:

```
1 {'ip': '100.0.0.0', 'longitude': 0.0, 'country_short': b'\x06\
    x10\x06\x04D', 'country_long': b'\x04D\x19L\x00A', 'latitude'
    : 0.0, 'city': b'This is a DB5 demo BIN database. You can
    evaluate IP address from 0.0.0.0 to 99.255.255.255.', 'region
    ': b'This is a DB5 demo BIN database. You can evaluate IP
    address from 0.0.0.0 to 99.255.255.255.'}
```

Příklad požadavku:

```
1 >>> import IP2Location
2 >>> ip2loc = IP2Location.IP2Location()
3 >>> ip2loc.open('IP-COUNTRY-REGION-CITY-LATITUDE-LONGITUDE-
    SAMPLE.BIN')
4 >>> print(ip2loc.get_all('8.8.8.8'))
```

Odpověď:

```
{'ip': '8.8.8.0', 'longitude': -122.078514, 'country_short': b'US', 'country_long': b'United States', 'latitude': 37.405991, 'city': b'Mountain View', 'region': b'California'}
```

Získávání geografických informací o IP adrese z databáze IP2Location probíhá obdobně jako u databáze MaxMind/GeoLite2City pomocí oficiálního modulu. Metoda `get` třídy `Ip2Location` (viz 4.6) požaduje, jako jeden z parametrů, cestu k souboru s geolokačními daty. Metoda načte databázový soubor, vyhledá IP adresu a data upraví do jednotné podoby. Poté jsou data vrácena ve formě objektu typu `IpLocation`.

```
class Ip2Location(IGeoIpDatabase):
    """
    Class for accessing geolocation data provided by IP2Location,
    available from https://www.ip2location.com/.
    Downloadable from http://lite.ip2location.com/database/ip-country-region-city-latitude-longitude.
    """

    @staticmethod
    def get(ip_address, api_key=None, db_path=None, username=None, password=None):
        # process request
        try:
            ip2loc = IP2Location.IP2Location()
            ip2loc.open(db_path)
        except:
            raise ServiceError()

        # content
        res = ip2loc.get_all(ip_address)

        if res is None:
            raise IpAddressNotFoundError(ip_address)

        # prepare return value
        ip_location = IpLocation(ip_address)

        # format data
        if res.country_short != ' ' \
           or res.country_short == 'N/A' \
           or res.country_short == '??':
            ip_location.country = res.country_short.decode('utf-8')
        else:
            ip_location.country = None

        if res.region != ' ' \
           or res.region == 'N/A':
            ip_location.region = res.region.decode('utf-8')
        else:
            ip_location.region = None

        if res.city != ' ' \
           or res.city == 'N/A':
```

```
42              ip_location.city = res.city.decode('utf-8')
43          else:
44              ip_location.city = None
45
46          if res.latitude != ' ' \
47             or res.latitude == 'N/A':
48              ip_location.latitude = float(res.latitude)
49          else:
50              ip_location.latitude = None
51
52          if res.longitude != ' ' \
53             or res.longitude == 'N/A':
54              ip_location.longitude = float(res.longitude)
55          else:
56              ip_location.longitude = None
57
58          return ip_location
```

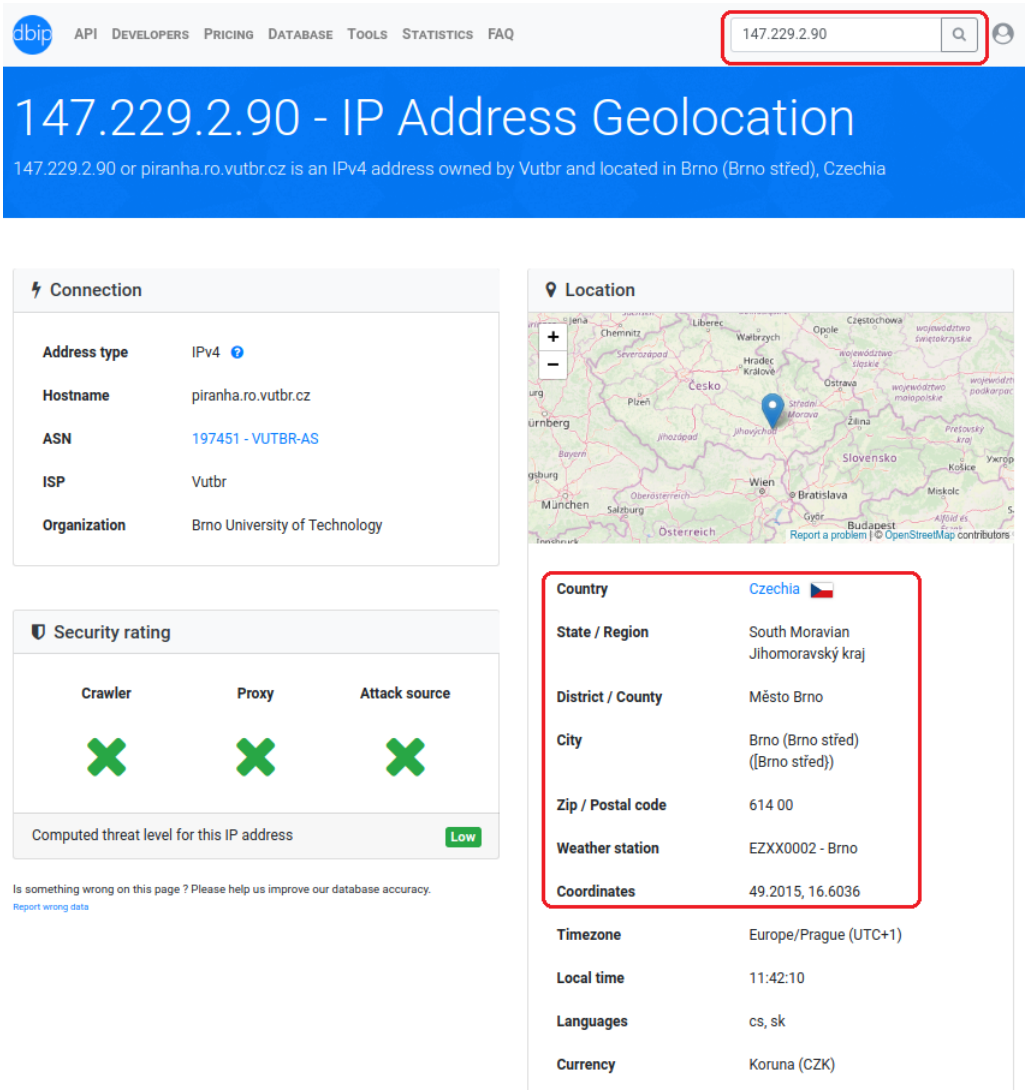
Výpis 4.6: Ukázka implementace třídy `Ip2Location`

### 4.4.3 Komerční databáze

#### DB-IP/IPtoLocation

Pokud máme pořízenou licenci ke geolokační databázi DB-IP, tak můžeme s výhodou využít třídy `DbIpCity` z modulu `ip2geotools.databases.noncommercial`, která pracuje přímo s API této služby. V případě, že licenci pořízenou nemáme, tak můžeme informace o geografické pozici IP adresy přímo vyhledávat na stránkách `https://db-ip.com/` (viz obr. 4.4), což má sloužit pro otestování plnohodnotné komerční geolokační databáze před jejím pořízením. Přímé ruční vyhledávání není efektivní při větším množství dat, a proto je výhodné použít vytvořenou třídu `DbIpWeb` zobrazenou na výpisu 4.7.

Metoda `get` této třídy nejprve zašle POST požadavek na server společnosti přesně v takovém formátu, jako bychom provedli ručně dotaz pomocí webového prohlížeče. Dále se provede kontrola, zda server vrátil regulérní odpověď HTTP 200 OK. Pokud ano, tak tělo odpovědi obsahuje zdrojový kód HTML (HyperText Markup Language) webové stránky určený k vykreslení webovým prohlížečem. Protože webové rozhraní je určené pouze k otestování komerční geolokační databáze, můžeme překročit povolené limity, a proto se ve webové stránce pokusíme o tom najít nějakou informaci. Pokud informaci nalezneme, vyhodíme výjimku, jinak pokračujeme ve zpracování. Nyní se dostáváme k nejnáročnější části implementace, a to jakým způsobem geolokační informace z webové stránky extrahovat. Z mého pohledu nejjednodušším a nejpřehlednějším způsobem je vyhledávání elementů v dokumentu pomocí CSS (kaskádové styly – Cascading Style Sheets) selektorů. Obdobného postupu využívá například také známá JavaScript knihovna jQuery[37]. A přesně tento




Obr. 4.4: Vyhledávání geolokačních informací o IP adrese na db-ip.com

způsob vyhledávání elementů v dokumentu nám umožní využití balíčku `pyquery`. Po vyhledání všech geolokačních informací třída vrátí objekt třídy `IpLocation` naplněný získanými daty.

```
1  class DbIpWeb(IGeoIpDatabase):
2      """
3      Class for accessing geolocation data provided by searching directly
4      on https://db-ip.com/.
5
6      """
7
8      @staticmethod
9      def get(ip_address, api_key=None, db_path=None, username=None, password=None):
10         # process request
11         try:
12             request = requests.post('https://db-ip.com/',
```

```
                                    headers={'User-Agent': 'Mozilla/5.0'},
                                    data=[('address', ip_address)],
                                    timeout=62)
        except:
            raise ServiceError()

        # check for HTTP errors
        if request.status_code != 200:
            raise ServiceError()

        # check for errors
        if b'you have exceeded the daily query limit' in request.content.lower():
            raise LimitExceededError()

        # parse content
        try:
            content = request.content.decode('utf-8')
            pq = pyquery.PyQuery(content)
            parsed_ip = pq('html > body div.container > h1') \
                        .remove('span') \
                        .text() \
                        .strip()
            parsed_country = pq('html > body > div.container table tr:contains("
                Country") td') \
                             .text() \
                             .strip()
            parsed_region = pq('html > body > div.container table tr:contains("
                State / Region") td') \
                            .text() \
                            .strip()
            parsed_city = pq('html > body > div.container table tr:contains("City")
                 td') \
                          .text() \
                          .strip()
            parsed_coords = pq('html > body > div.container table tr:contains("
                Coordinates") td') \
                            .text() \
                            .strip() \
                            .split(',')
        except:
            raise InvalidResponseError()

        # check for errors
        if ip_address != parsed_ip:
            raise IpAddressNotFoundError(ip_address)

        # prepare return value
        ip_location = IpLocation(ip_address)

        # format data
        try:
            ip_location.country = parsed_country
            ip_location.region = parsed_region
            ip_location.city = parsed_city
            ip_location.latitude = float(parsed_coords[0].strip())
            ip_location.longitude = float(parsed_coords[1].strip())
        except:
            ip_location.country = None
            ip_location.region = None
```

```
68              ip_location.city = None
69              ip_location.latitude = None
70              ip_location.longitude = None
71
72          return ip_location
```

Výpis 4.7: Ukázka implementace třídy `DbIpWeb`

### MaxMind/GeoIP2City

Společnost MaxMind poskytuje velmi kvalitní webové API pod názvem „GeoIP2 Precision Web Services“, ke kterému je možno přistoupit pomocí knihoven dostupných pro velkou škálu programovacích jazyků. V případě jazyka Python se nabízí využít oficiální balíček `geoip2`, který pro naše využití neposkytuje jedinou avšak velmi zásadní funkcionalitu, a to přístup k webovému API v „demo“ režimu. Z tohoto důvodu jsem při vytváření třídy `MaxMindGeoIp2City` musel provést napojení na poskytované rozhraní kompletně od začátku. Demo režim funguje takto:

Příklad požadavku:

```
1 https://www.maxmind.com/geoip/v2.1/city/147.229.2.90?demo=1
```

Odpověď:

```
1  {
2    "city": {
3      "geoname_id": 3078610,
4      "names": {
5        "en": "Brno"
6      }
7    },
8    "continent": {
9      "code": "EU",
10     "geoname_id": 6255148,
11     "names": {
12       "en": "Europe"
13     }
14   },
15   "country": {
16     "is_in_european_union": true,
17     "iso_code": "CZ",
18     "geoname_id": 3077311,
19     "names": {
20       "en": "Czechia"
```

```
    }
  },
  "location": {
    "accuracy_radius": 5,
    "latitude": 49.2333,
    "longitude": 16.65,
    "time_zone": "Europe/Prague"
  },
  "postal": {
    "code": "614 00"
  },
  "registered_country": {
    "is_in_european_union": true,
    "iso_code": "CZ",
    "geoname_id": 3077311,
    "names": {
      "en": "Czechia"
    }
  },
  "subdivisions": [
    {
      "iso_code": "64",
      "geoname_id": 3339536,
      "names": {
        "en": "South Moravian"
      }
    },
    {
      "iso_code": "642",
      "geoname_id": 3078609,
      "names": {
        "en": "Mesto Brno"
      }
    }
  ],
  "traits": {
    "autonomous_system_number": 197451,
    "autonomous_system_organization": "Brno University of
       Technology",
    "domain": "vutbr.cz",
    "isp": "Brno University of Technology",
```

```
61      "organization": "Brno University of Technology",
62      "ip_address": "147.229.2.90"
63    }
64  }
```

Třída `MaxMindGeoIp2City` (ukázka viz 4.8) umožňuje pracovat ve dvou režimech – placený přístup i demo verze. V případě, že máme zakoupenou komerční licenci a známe své přístupové údaje, můžeme je využít. Pokud je nemáme/neznáme, dotazy budou fungovat v demo režimu s omezeným počtem dotazů. API poskytuje odpovědi ve formátu JSON při dodržení principů návrhu REST, což nám umožňuje vyhazovat velmi konkrétní výjimky. V případě, že nedošlo k žádnému problému, JSON odpověď se převede na objekt, patřičně zpracuje a výsledkem je objekt typu `IpLocation` naplněný daty.

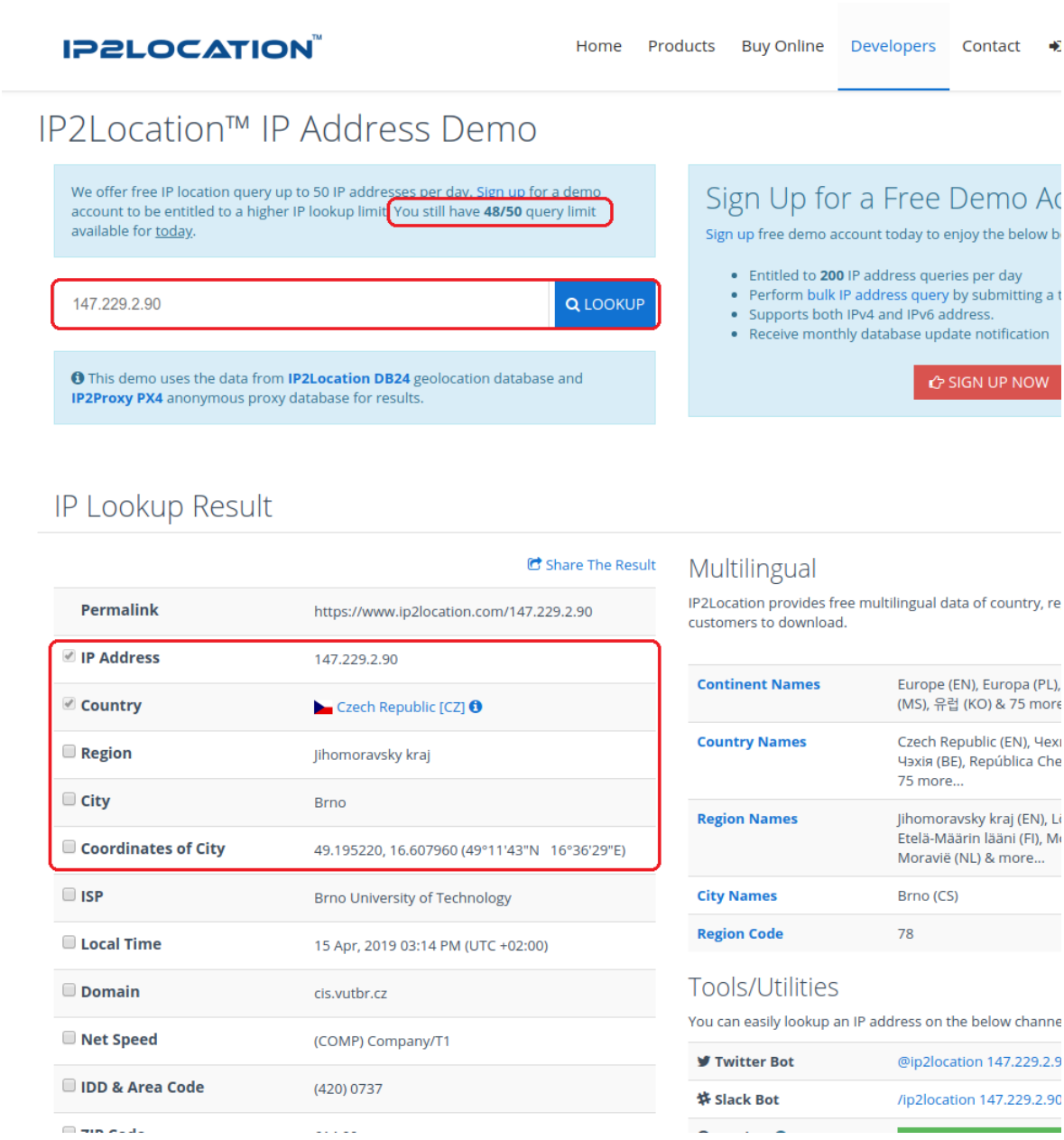
```
1  class MaxMindGeoIp2City(IGeoIpDatabase):
2      """
3      Class for accessing geolocation data provided by GeoIP2 database
4      created by MaxMind, available from https://www.maxmind.com/.
5      """
6
7      @staticmethod
8      def get(ip_address, api_key=None, db_path=None, username=None, password=None):
9          # process request
10         try:
11             # optional auth for increasing amount of queries per day
12             if username != None and password != None:
13                 auth = HTTPBasicAuth(username, password)
14             else:
15                 auth = None
16
17             request = requests.get('https://www.maxmind.com/geoip/v2.1/city/'
18                                    + quote(ip_address)
19                                    + ('?demo=1' if auth == None else ''),
20                                    auth=auth,
21                                    timeout=62)
22         except:
23             raise ServiceError()
24
25         # parse content
26         try:
27             content = request.content.decode('utf-8')
28             content = json.loads(content)
29         except:
30             raise InvalidResponseError()
31
32         # check for HTTP errors
33         if request.status_code != 200:
34             if request.status_code == 400:
35                 raise InvalidRequestError(content['code'])
36             elif request.status_code == 401:
37                 raise PermissionRequiredError(content['code'])
38             elif request.status_code == 402:
39                 raise LimitExceededError(content['code'])
40             elif request.status_code == 403:
```

```
41                  raise PermissionRequiredError(content['code'])
42              elif request.status_code == 404:
43                  raise IpAddressNotFoundError(ip_address)
44              elif request.status_code == 500:
45                  raise InvalidRequestError()
46              else:
47                  raise ServiceError()
48
49          # prepare return value
50          ip_location = IpLocation(ip_address)
51
52          # format data
53          if content.get('country'):
54              if content['country'].get('iso_code'):
55                  ip_location.country = content['country']['iso_code']
56              else:
57                  ip_location.country = None
58          else:
59              ip_location.country = None
60
61          if content.get('subdivisions'):
62              if content['subdivisions'][0].get('names'):
63                  if content['subdivisions'][0]['names'].get('en'):
64                      ip_location.region = content['subdivisions'][0]['names']['en']
65                  else:
66                      ip_location.region = None
67              else:
68                  ip_location.region = None
69          else:
70              ip_location.region = None
71
72          if content.get('city'):
73              if content['city'].get('names'):
74                  if content['city']['names'].get('en'):
75                      ip_location.city = content['city']['names']['en']
76                  else:
77                      ip_location.city = None
78              else:
79                  ip_location.city = None
80          else:
81              ip_location.city = None
82
83          if content.get('location'):
84              ip_location.latitude = float(content['location']['latitude'])
85              ip_location.longitude = float(content['location']['longitude'])
86          else:
87              ip_location.latitude = None
88              ip_location.longitude = None
89
90          return ip_location
```

Výpis 4.8: Ukázka implementace třídy `MaxMindGeoIp2City`

### IP2Location/DB24

Databáze IP2Location poskytuje „demo“ své nejvyšší verze geolokační databáze přístupné pomocí přímého vyhledávání informací o IP adresách na stránkách `https://www.ip2location.com/demo` (což můžeme vidět na obr. 4.5). Třída `Ip2LocationWeb` (viz 4.9), resp. její metoda `get` funguje na zcela odlišném principu než u ostatních databází.




Obr. 4.5: Vyhledávání geolokačních informací o IP adrese na ip2location.com

Metoda pro získávání informací o IP adrese využívá na pozadí internetový pro-

hlížeč Mozilla Firefox spouštěný pomocí knihovny `selenium`. Toto řešení bylo nutné využít z důvodu zavedení ochrany proti automatizovanému zpracování dat ze strany společnosti IP2Location použitím ochrany Google reCAPTCHA v její nejjednodušší podobě. Metoda `get` pomocí knihovny `selenium` spustí internetový prohlížeč s URL adresou, na které společnost poskytuje demo své geolokační databáze. Následně nám knihovna zprostředkuje strukturu webové stránky, kterou je možné parsovat obvyklými způsoby. První ověřovanou informací je aktuální volný limit na počet dotazů, který v současné době činí 50 dotazů za den. Jestliže máme volné jednotky na dotazy, tak proběhne vyhledání elementů ve webové stránce, jejich separace, úprava a vrácení ve formě objektu `IpLocation`.

```
class Ip2LocationWeb(IGeoIpDatabase):
    """
    Class for accessing geolocation data provided by searching directly
    on https://www.ip2location.com/.

    """

    @staticmethod
    def get(ip_address, api_key=None, db_path=None, username=None, password=None):
        # initiate headless Firefox using selenium to pass through Google reCAPTCHA
        options = Options()
        options.headless = True
        browser = webdriver.Firefox(options=options)

        try:
            browser.get('http://www.ip2location.com/demo/' + ip_address)
            element = WebDriverWait(browser, 30).until(
                EC.presence_of_element_located((By.NAME, 'ipAddress'))
            )

            if not element:
                raise Exception
        except:
            raise ServiceError()

        # parse current limit
        current_limit = 0
        body = browser.find_element_by_tag_name('body').text

        try:
            limit = re.search(r'You still have.*?([\d]{1,2})/50.* query limit',
                              body,
                              re.DOTALL)

            if limit != None:
                current_limit = int(limit.group(1))
        except:
            raise InvalidResponseError()

        # check if limit is exceeded
        if current_limit == 0:
            raise LimitExceededError()

        # parse content
```

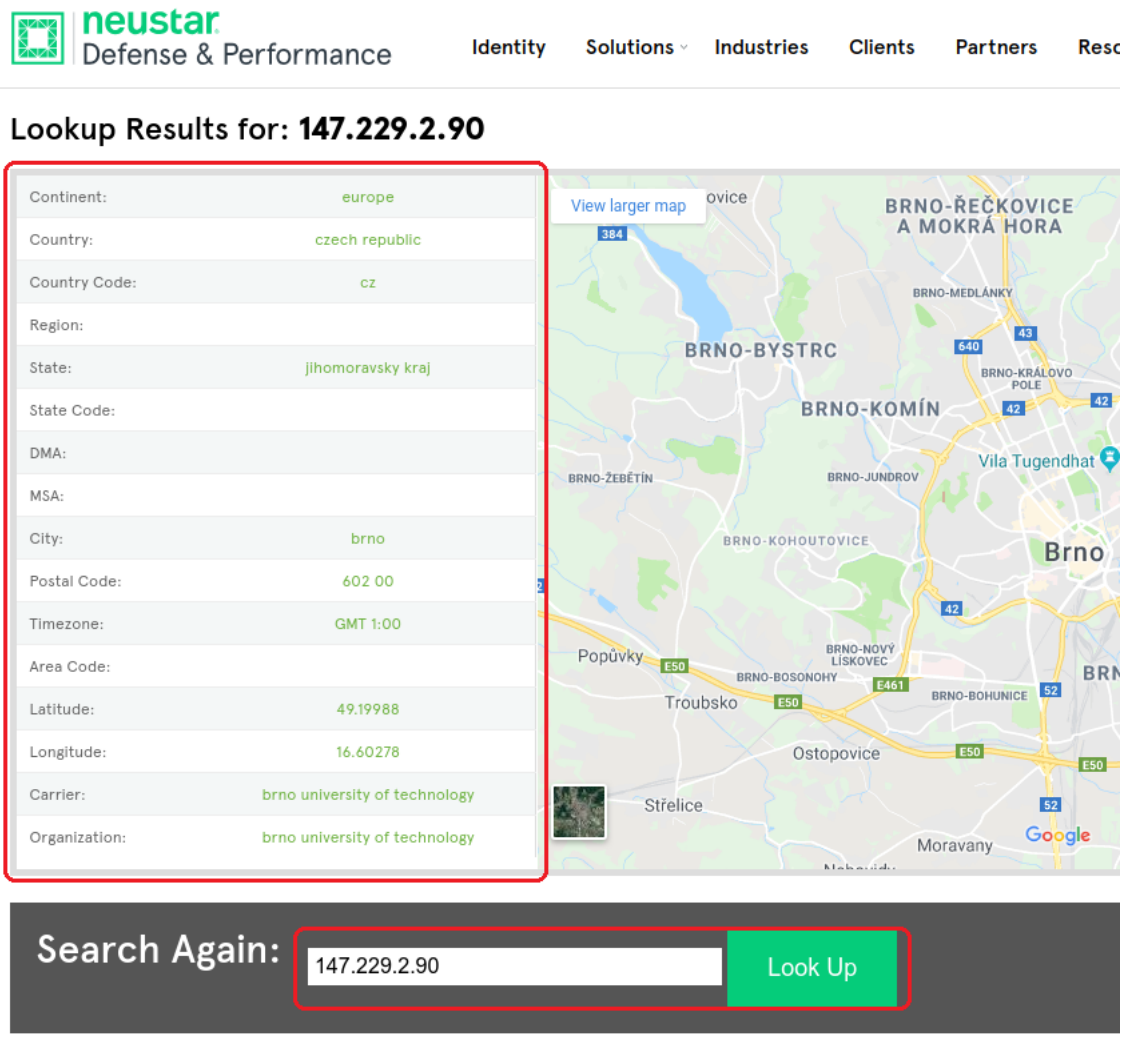
```
45          try:
46              table = browser.find_element_by_xpath('//table[contains(.,"Permalink")]
                    ')
47
48              parsed_ip = table.find_element_by_xpath('//tr[contains(.,"IP Address")
                    ]/td').text.strip()
49              parsed_country = [class_name.replace('flag-icon-', '').upper() for
                    class_name in table.find_element_by_class_name('flag-icon').
                    get_attribute('class').split(' ') if class_name.startswith('flag-
                    icon-')][0]
50              parsed_region = table.find_element_by_xpath('//tr[contains(.,"Region")
                    ]/td').text.strip()
51              parsed_city = table.find_element_by_xpath('//tr[contains(.,"City")]/td'
                    ).text.strip()
52              parsed_coords = table.find_element_by_xpath('//tr[contains(.,"
                    Coordinates of City")]/td').text.strip()
53          except:
54              raise InvalidResponseError()
55
56          # exit headless firefox
57          browser.quit()
58
59          # check for errors
60          if ip_address != parsed_ip:
61              raise IpAddressNotFoundError(ip_address)
62
63          # prepare return value
64          ip_location = IpLocation(ip_address)
65
66          # format data
67          try:
68              ip_location.country = parsed_country
69              ip_location.region = parsed_region
70              ip_location.city = parsed_city
71
72              parsed_coords = parsed_coords.split('(')[0].split(',')
73              ip_location.latitude = float(parsed_coords[0].strip())
74              ip_location.longitude = float(parsed_coords[1].strip())
75          except:
76              ip_location.country = None
77              ip_location.region = None
78              ip_location.city = None
79              ip_location.latitude = None
80              ip_location.longitude = None
81
82          return ip_location
```

Výpis 4.9: Ukázka implementace třídy `Ip2LocationWeb`

### Neustar

Ke geolokačním informacím z databáze Neustar se taktéž přistupuje způsobem, kdy je nutné napodobit manuální vyhledávání ve webovém prohlížeči pomocí požadavku POST náležitého formátu. Vyhledávání a jeho výsledky jsou znázorněny na obr. 4.6.

Třída `NeustarWeb` (viz 4.10) a její metoda `get` pracuje s parsováním webové




Obr. 4.6: Vyhledávání geolokačních informací o IP adrese na www.home.neustar

stránky (podobně jako u předchozí databáze). Nejprve se provede POST požadavek na server geolokační databáze, ověří se vrácená data (příp. jsou vyhozené výjimky) a díky modulu `pyquery` jsou informace o IP adrese dohledány ve webové stránce. Následně jsou nalezené informace vráceny ve formě naplněného objektu třídy `IpLocation`.

```
1  class NeustarWeb(IGeoIpDatabase):
2      """
3      Class for accessing geolocation data provided by searching directly
4      on https://www.home.neustar/resources/tools/ip-geolocation-lookup-tool/.
5
6      """
7
8      @staticmethod
9      def get(ip_address, api_key=None, db_path=None, username=None, password=None):
10         # process request
```

```
11          try:
12              request = requests.post('https://www.home.neustar/resources/tools/ip-
                    geolocation-lookup-tool',
13                                      headers={'User-Agent': 'Mozilla/5.0'},
14                                      data=[('ip', ip_address)],
15                                      timeout=62)
16          except:
17              raise ServiceError()
18
19          # check for HTTP errors
20          if request.status_code != 200:
21              raise ServiceError()
22
23          # check for errors
24          if b'rate limit exceeded' in request.content.lower():
25              raise LimitExceededError()
26
27          # parse content
28          try:
29              content = request.content.decode('utf-8')
30              pq = pyquery.PyQuery(content)
31              parsed_ip = pq('html > body > section.full.resource article h2 > strong
                    ') \
32                      .text() \
33                      .strip()
34              parsed_country = pq('html > body > section.full.resource article div.
                    data >table:first tr:contains("Country Code:") td:not(.item)') \
35                          .text() \
36                          .strip() \
37                          .upper()
38              parsed_region = pq('html > body > section.full.resource article div.
                    data >table:first tr:contains("Region:") td:not(.item)') \
39                          .text() \
40                          .strip() \
41                          .title()
42              parsed_state = pq('html > body > section.full.resource article div.data
                     >table:first tr:contains("State:") td:not(.item)') \
43                          .text() \
44                          .strip() \
45                          .title()
46              parsed_city = pq('html > body > section.full.resource article div.data
                    >table:first tr:contains("City:") td:not(.item)') \
47                          .text() \
48                          .strip() \
49                          .title()
50              parsed_latitude = pq('html > body > section.full.resource article div.
                    data >table:first tr:contains("Latitude:") td:not(.item)') \
51                          .text() \
52                          .strip()
53              parsed_longitude = pq('html > body > section.full.resource article div.
                    data >table:first tr:contains("Longitude:") td:not(.item)') \
54                          .text() \
55                          .strip()
56          except:
57              raise InvalidResponseError()
58
59          # check for errors
60          if ip_address != parsed_ip:
61              raise IpAddressNotFoundError(ip_address)
```

```
62
63          # prepare return value
64          ip_location = IpLocation(ip_address)
65
66          # format data
67          try:
68              ip_location.country = parsed_country
69
70              if parsed_region is None:
71                  ip_location.region = parsed_region
72              else:
73                  ip_location.region = parsed_state
74
75              ip_location.city = parsed_city
76              ip_location.latitude = float(parsed_latitude)
77              ip_location.longitude = float(parsed_longitude)
78          except:
79              ip_location.country = None
80              ip_location.region = None
81              ip_location.city = None
82              ip_location.latitude = None
83              ip_location.longitude = None
84
85          return ip_location
```

Výpis 4.10: Ukázka implementace třídy `NeustarWeb`

### Geobytes

API Geobytes Get City Details patří mezi ta nejjednodušší rozhraní. Rozhraní nám ani neposkytuje detailnější informace o nastalých chybách, a proto není možné je detailněji rozlišovat.

Příklad požadavku:

```
1 http://getcitydetails.geobytes.com/GetCityDetails?fqcn
      =147.229.2.90
```

Odpověď:

```
1  {
2    "geobytesforwarderfor": "",
3    "geobytesremoteip": "83.240.64.230",
4    "geobytesipaddress": "147.229.2.90",
5    "geobytescertainty": "99",
6    "geobytesinternet": "CZ",
7    "geobytescountry": "Czech Republic",
8    "geobytesregionlocationcode": "CZJM",
9    "geobytesregion": "Jihomoravsky Kraj",
10   "geobytescode": "JM",
```

```
11    "geobyteslocationcode": "CZJMBRNO",
12    "geobytesdma": "0",
13    "geobytescity": "Brno",
14    "geobytescityid": "3639",
15    "geobytesfqcn": "Brno, JM, Czech Republic",
16    "geobyteslatitude": "49.200001",
17    "geobyteslongitude": "16.632999",
18    "geobytescapital": "Prague",
19    "geobytestimezone": "+01:00",
20    "geobytesnationalitysingular": "Czech",
21    "geobytespopulation": "10264212",
22    "geobytesnationalityplural": "Czechs",
23    "geobytesmapreference": "Europe ",
24    "geobytescurrency": "Czech Koruna",
25    "geobytescurrencycode": "CZK",
26    "geobytestitle": "The Czech Republic"
27 }
```

Implementace této geolokační databáze do třídy `GeobytesCityDetails` je jednoduchá a přímočará. Ukázka je zobrazena na výpisu 4.11. Metoda `get` nejdříve zašle GET požadavek na patřičnou URL adresu obsahující jako parametr IP adresu a získá odpověď ve formátu JSON, která je převedena na objekty. Vybrané údaje jsou vráceny ve formě objektu `IpLocation`.

```
1  class GeobytesCityDetails(IGeoIpDatabase):
2      """
3      Class for accessing geolocation data provided by
4      http://geobytes.com/get-city-details-api/.
5      """
6
7      @staticmethod
8      def get(ip_address, api_key=None, db_path=None, username=None, password=None):
9          # process request
10         try:
11             request = requests.get('http://getcitydetails.geobytes.com/
                   GetCityDetails?fqcn='
12                                    + quote(ip_address),
13                                    timeout=62)
14         except:
15             raise ServiceError()
16
17         # check for HTTP errors
18         if request.status_code != 200:
19             raise ServiceError()
20
21         # parse content
22         try:
23             content = request.content.decode('latin-1')
24             content = json.loads(content)
```

```
25          except:
26              raise InvalidResponseError()
27
28          # prepare return value
29          ip_location = IpLocation(ip_address)
30
31          # format data
32          if content.get('geobytesinternet'):
33              ip_location.country = content['geobytesinternet']
34          else:
35              ip_location.country = None
36
37          if content.get('geobytesregion'):
38              ip_location.region = content['geobytesregion']
39          else:
40              ip_location.region = None
41
42          if content.get('geobytescity'):
43              ip_location.city = content['geobytescity']
44          else:
45              ip_location.city = None
46
47          if content.get('geobyteslatitude') and content.get('geobyteslongitude'):
48              ip_location.latitude = float(content['geobyteslatitude'])
49              ip_location.longitude = float(content['geobyteslongitude'])
50          else:
51              ip_location.latitude = None
52              ip_location.longitude = None
53
54          return ip_location
```

Výpis 4.11: Ukázka implementace třídy `GeobytesCityDetails`

### Skyhook

Geolokační databáze Skyhook má API velmi podobné jako databáze předchozí. Poměrně zvláštním způsobem jsou ale ošetřeny chybové stavy, kdy část chybových stavů je řešena na úrovni HTTP stavových kódů (401 Unauthorized, 500 Internal Server Error) a část je řešena na úrovni množství dat, kdy téměř prázdná odpověď ze serveru znamená, že IP adresa nebyla nalezena.

Příklad požadavku:

```
1 https://context.skyhookwireless.com/accelerator/ip?ip
      =147.229.2.90&user=eval&key=API_KLIC&version=2.0&prettyPrint
      =true
```

Odpověď:

```
1 {
2   "data": {
```

```
3       "location": {
4         "type": "FIXED",
5         "latitude": 49.285160064697266,
6         "longitude": 16.631999969482422,
7         "hpe": 202448
8       },
9       "ip": "147.229.2.90",
10      "civic": {
11        "country": "Czech Republic",
12        "countryProb": 1,
13        "countryIso": "CZ"
14      }
15    }
16  }
```

Třída, která má získávat geolokační data z této databáze, je pojmenována poměrně dlouhým názvem – `SkyhookContextAcceleratorIp` (viz 4.12). Pro úspěšné napojení na tuto databázi jsou nutné 2 údaje, a to uživatelské jméno a heslo. Uživatelské jméno je na straně API stejné pro všechny („eval“), nicméně předpokládám, že tato situace se může v budoucnu změnit. Metoda `get` nejprve zašle GET požadavek na API, následně jsou zkontrolovány HTTP stavové kódy a příp. vyhozeny odpovídající výjimky. Dále se ještě provede další kontrola, zda byla IP adresa nalezena. Poté se provede ověření, které informace byly získány, a vrátí se objekt typu `IpLocation` plný dat.

```
1  class SkyhookContextAcceleratorIp(IGeoIpDatabase):
2      """
3      Class for accessing geolocation data provided by http://www.skyhookwireless.com
           /.
4
5      """
6
7      @staticmethod
8      def get(ip_address, api_key=None, db_path=None, username=None, password=None):
9          # process request
10         try:
11             request = requests.get('https://context.skyhookwireless.com/accelerator
                   /ip?'
12                                    + 'ip=' + quote(ip_address)
13                                    + '&user=' + quote(username)
14                                    + '&key=' + quote(password)
15                                    + '&version=2.0',
16                                    timeout=62)
17         except:
18             raise ServiceError()
19
20         # check for HTTP errors
21         if request.status_code != 200:
22             if request.status_code == 400:
```

```
23                  raise InvalidRequestError()
24              elif request.status_code == 401:
25                  raise PermissionRequiredError(ip_address)
26              else:
27                  raise ServiceError()
28
29          # content decode
30          try:
31              content = request.content.decode('utf-8')
32          except:
33              raise InvalidResponseError()
34
35          # check for IP address not found error
36          if content == '{"data":{"ip":"' + ip_address + '"}}':
37              raise IpAddressNotFoundError(ip_address)
38
39          # parse content
40          try:
41              content = json.loads(content)
42          except:
43              raise InvalidResponseError()
44
45          # prepare return value
46          ip_location = IpLocation(ip_address)
47
48          # format data
49          if content.get('data'):
50              if content['data'].get('civic'):
51                  if content['data']['civic'].get('countryIso'):
52                      ip_location.country = content['data']['civic']['countryIso']
53                  else:
54                      ip_location.country = None
55
56                  if content['data']['civic'].get('state'):
57                      ip_location.region = content['data']['civic']['state']
58                  else:
59                      ip_location.region = None
60
61                  if content['data']['civic'].get('city'):
62                      ip_location.city = content['data']['civic']['city']
63                  else:
64                      ip_location.city = None
65              else:
66                  ip_location.country = None
67                  ip_location.region = None
68                  ip_location.city = None
69
70              if content['data'].get('location'):
71                  if content['data']['location'].get('latitude') \
72                     and content['data']['location'].get('longitude'):
73                      ip_location.latitude = content['data']['location']['latitude']
74                      ip_location.longitude = content['data']['location']['longitude'
                            ]
75                  else:
76                      ip_location.latitude = None
77                      ip_location.longitude = None
78              else:
79                  ip_location.latitude = None
80                  ip_location.longitude = None
```

```
81          else:
82              ip_location.country = None
83              ip_location.region = None
84              ip_location.city = None
85              ip_location.latitude = None
86              ip_location.longitude = None
87
88          return ip_location
```

Výpis 4.12: Ukázka implementace třídy `SkyhookContextAcceleratorIp`

### ipinfo

Přístup ke geolokačním datům databáze ipinfo je velmi jednoduchý díky jejich webovému API respektující obecné zvyklosti a přístup REST.

Příklad požadavku:

```
1 https://ipinfo.io/147.229.2.90/json
```

Odpověď:

```
1  {
2    "ip": "147.229.2.90",
3    "hostname": "piranha.ro.vutbr.cz",
4    "city": "Brno",
5    "region": "South Moravian",
6    "country": "CZ",
7    "loc": "49.2000,16.6333",
8    "org": "AS197451 Brno University of Technology",
9    "postal": "614 00"
10 }
```

Metoda `get` třídy `IpInfo` (viz 4.13) zašle GET požadavek na server ipinfo.io, kde je jako parametr uvedena IP adresa a dále je zde použit parametr „geo“, kterým omezíme množství dat, které požadujeme. Odpověď ze serveru je zkontrolována na chyby, a to především na chybu vzniklou překročením limitu na dotazy na tuto službu. Odpověď ve formátu JSON je převedena na objekt typu `IpLocation`.

```
1  class IpInfo(IGeoIpDatabase):
2      """
3      Class for accessing geolocation data provided by https://ipinfo.io/.
4      """
5
6      @staticmethod
7      def get(ip_address, api_key=None, db_path=None, username=None, password=None):
8          # process request
9          try:
```

```
10              request = requests.get('https://ipinfo.io/' + quote(ip_address) + '/geo
                    /',
11                                      timeout=62)
12          except:
13              raise ServiceError()
14
15          # check for HTTP errors
16          if request.status_code != 200:
17              if request.status_code == 404:
18                  raise IpAddressNotFoundError(ip_address)
19              elif request.status_code == 429:
20                  raise LimitExceededError()
21              elif request.status_code == 500:
22                  raise InvalidRequestError()
23              else:
24                  raise ServiceError()
25
26          # parse content
27          try:
28              content = request.content.decode('utf-8')
29              content = json.loads(content)
30          except:
31              raise InvalidResponseError()
32
33          # prepare return value
34          ip_location = IpLocation(ip_address)
35
36          # format data
37          if content.get('country'):
38              ip_location.country = content['country']
39          else:
40              ip_location.country = None
41
42          if content.get('region'):
43              ip_location.region = content['region']
44          else:
45              ip_location.region = None
46
47          if content.get('city'):
48              ip_location.city = content['city']
49          else:
50              ip_location.city = None
51
52          if content.get('loc'):
53              location = content['loc'].split(',')
54              ip_location.latitude = float(location[0])
55              ip_location.longitude = float(location[1])
56          else:
57              ip_location.latitude = None
58              ip_location.longitude = None
59
60          return ip_location
```

Výpis 4.13: Ukázka implementace třídy `IpInfo`

### EurekAPI

API této geolokační služby umožňuje získávat v několika formátech, a to XML a JSON, resp. také jako JSONP.

(a) XML

Příklad požadavku:

```
http://api.eurekapi.com/iplocation/v1.8/locateip?key=
    API_KLIC&ip=147.229.2.90&format=XML&compact=N
```

Odpověď:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <query_status>
    <query_status_code>OK</query_status_code>
    <query_status_description>Query successfully performed.</
        query_status_description>
  </query_status>
  <ip_address>147.229.2.90</ip_address>
  <geolocation_data>
    <continent_code>EU</continent_code>
    <continent_name>Europe</continent_name>
    <country_code_iso3166alpha2>CZ</
        country_code_iso3166alpha2>
    <country_code_iso3166alpha3>CZE</
        country_code_iso3166alpha3>
    <country_code_iso3166numeric>203</
        country_code_iso3166numeric>
    <country_code_fips10-4>EZ</country_code_fips10-4>
    <country_name>Czech Republic</country_name>
    <region_code>EZ78</region_code>
    <region_name>Jihomoravsky kraj</region_name>
  </geolocation_data>
</response>
```

(b) JSON

Příklad požadavku:

```
http://api.eurekapi.com/iplocation/v1.8/locateip?key=
    API_KLIC&ip=147.229.2.90&format=JSON&compact=N
```

Odpověď:

```
{
  "query_status": {
    "query_status_code": "OK",
    "query_status_description": "Query successfully
       performed."
  },
  "ip_address": "147.229.2.90",
  "geolocation_data": {
    "continent_code": "EU",
    "continent_name": "Europe",
    "country_code_iso3166alpha2": "CZ",
    "country_code_iso3166alpha3": "CZE",
    "country_code_iso3166numeric": "203",
    "country_code_fips10-4": "EZ",
    "country_name": "Czech Republic",
    "region_code": "EZ78",
    "region_name": "Jihomoravsky kraj"
  }
}
```

Napojení na tuto geolokační službu zajišťuje třída `Eurek` (viz 4.14) s metodou `get`, která zašle požadavek na API a očekává odpověď ve formátu JSON. Dle doporučení této služby je prováděna kontrola na HTTP stavový kód 429 Too Many Requests. Rozhraní používá pro sdělování chybových stavů poměrně nešťastné řešení, a to odpověď s HTTP stavovým kódem 200 OK, kde odpověď obsahuje informaci o našem požadavku, resp. zda došlo k nějaké chybě, což jsem převedl na vyhazování odpovídajících výjimek. Nakonec jsou informace z odpovědi extrahovány a vráceny ve formě objektu třídy `IpLocation`.

```
class Eurek(IGeoIpDatabase):
    """
    Class for accessing geolocation data provided by https://www.eurekapi.com/.
    """

    @staticmethod
    def get(ip_address, api_key=None, db_path=None, username=None, password=None):
        # process request
        try:
            request = requests.get('https://https-api.eurekapi.com/iplocation/v1.8/
                locateip?'
                                   + 'ip=' + quote(ip_address)
                                   + '&key=' + quote(api_key)
                                   + '&format=JSON',
                                   timeout=62)
        except:
```

```
16              raise ServiceError()
17
18          # check for HTTP errors
19          if request.status_code != 200:
20              if request.status_code == 429:
21                  raise LimitExceededError()
22              elif request.status_code == 500:
23                  raise InvalidRequestError()
24              else:
25                  raise ServiceError()
26
27          # parse content
28          try:
29              content = request.content.decode('utf-8')
30              content = json.loads(content)
31          except:
32              raise InvalidResponseError()
33
34          # prepare return value
35          ip_location = IpLocation(ip_address)
36
37          # check for errors
38          if content['query_status']['query_status_code'] != 'OK':
39              error_status = content['query_status']['query_status_code']
40              error_status_desc = content['query_status']['query_status_description']
41
42              if error_status == 'MISSING_SERVICE_ACCESS_KEY' \
43                 or error_status == 'INVALID_SERVICE_ACCESS_KEY' \
44                 or error_status == 'FREE_TRIAL_LICENSE_EXPIRED' \
45                 or error_status == 'SUBSCRIPTION_EXPIRED':
46                  raise PermissionRequiredError(error_status_desc)
47              elif error_status == 'MISSING_IP_ADDRESS' \
48                   or error_status == 'INVALID_IP_ADDRESS':
49                  raise IpAddressNotFoundError(ip_address)
50              else:
51                  ip_location.country = None
52                  ip_location.region = None
53                  ip_location.city = None
54                  ip_location.latitude = None
55                  ip_location.longitude = None
56                  return ip_location
57
58          # format data
59          if content.get('geolocation_data'):
60              if content['geolocation_data'].get('country_code_iso3166alpha2'):
61                  ip_location.country = content['geolocation_data']['
                        country_code_iso3166alpha2']
62              else:
63                  ip_location.country = None
64
65              if content['geolocation_data'].get('region_name'):
66                  ip_location.region = content['geolocation_data']['region_name']
67              else:
68                  ip_location.region = None
69
70              if content['geolocation_data'].get('city'):
71                  ip_location.city = content['geolocation_data']['city']
72              else:
73                  ip_location.city = None
```

```
74
75              if content['geolocation_data'].get('latitude') \
76                 and content['geolocation_data'].get('longitude'):
77                  ip_location.latitude = float(content['geolocation_data']['latitude'
                        ])
78                  ip_location.longitude = float(content['geolocation_data']['
                        longitude'])
79              else:
80                  ip_location.latitude = None
81                  ip_location.longitude = None
82          else:
83              ip_location.country = None
84              ip_location.region = None
85              ip_location.city = None
86              ip_location.latitude = None
87              ip_location.longitude = None
88
89          return ip_location
```

Výpis 4.14: Ukázka implementace třídy `Eurek`

### ipdata

Tato geolokační služba má velmi jednoduché API umožňující získávat data pouze ve formátu JSON.

(a) JSON

Příklad požadavku:

```
1 https://api.ipdata.co/147.229.2.90?api-key=API_KLIC
```

Odpověď:

```
1  {
2    "ip": "147.229.2.90",
3    "is_eu": true,
4    "city": "Brno",
5    "region": "South Moravian",
6    "region_code": "64",
7    "country_name": "Czechia",
8    "country_code": "CZ",
9    "continent_name": "Europe",
10   "continent_code": "EU",
11   "latitude": 49.2333,
12   "longitude": 16.65,
13   "asn": "AS197451",
14   "organisation": "Brno University of Technology",
15   "postal": "614 00",
16   "calling_code": "420",
```

```
  "flag": "https://ipdata.co/flags/cz.png",
  "emoji_flag": "xx",
  "emoji_unicode": "U+1F1E8 U+1F1FF",
  "languages": [
    {
      "name": "Czech",
      "native": "Česky"
    },
    {
      "name": "Slovak",
      "native": "Slovenčina"
    }
  ],
  "currency": {
    "name": "Czech Republic Koruna",
    "code": "CZK",
    "symbol": "Kč",
    "native": "Kč",
    "plural": "Czech Republic korunas"
  },
  "time_zone": {
    "name": "Europe/Prague",
    "abbr": "CET",
    "offset": "+0100",
    "is_dst": false,
    "current_time": "2019-02-19T21:20:28.064080+01:00"
  },
  "threat": {
    "is_tor": false,
    "is_proxy": false,
    "is_anonymous": false,
    "is_known_attacker": false,
    "is_known_abuser": false,
    "is_threat": false,
    "is_bogon": false
  }
}
```


Třída `Ipdata` zajišťující napojení na tuto geolokační službu (viz 4.15) obsahuje metodu `get`, která zašle požadavek na API a očekává odpověď ve formátu JSON. Následně dochází ke kontrole HTTP stavového kódu, který je příp. převeden na vy-

hození patřičné výjimky – API může odpovědět celkem až 4 různými stavovými kódy (200 OK, 400 Bad Request, 401 Unauthorized nebo 403 Forbidden). Kód 400 obsahuje v odpovědi doplňující informace o vzniklé chybě. Nakonec jsou z odpovědi extrahovány patřičné informace a vráceny ve formě objektu třídy `IpLocation`.

```
class Ipdata(IGeoIpDatabase):
    """
    Class for accessing geolocation data provided by https://ipdata.co/.
    """

    @staticmethod
    def get(ip_address, api_key='test', db_path=None, username=None, password=None)
        :
        # process request
        try:
            request = requests.get('https://api.ipdata.co/' + quote(ip_address)
                                   + '?api-key=' + quote(api_key),
                                   timeout=62)
        except:
            raise ServiceError()

        # check for HTTP errors
        if request.status_code != 200 and request.status_code != 400:
            if request.status_code == 401:
                raise PermissionRequiredError()
            elif request.status_code == 403:
                raise LimitExceededError()
            else:
                raise ServiceError()

        # parse content
        try:
            content = request.content.decode('utf-8')
            content = json.loads(content)
        except:
            raise InvalidResponseError()

        # check for errors
        if content.get('message'):
            if 'private IP address' in content['message']:
                raise IpAddressNotFoundError(ip_address)
            else:
                raise InvalidRequestError()

        # prepare return value
        ip_location = IpLocation(ip_address)

        # format data
        if content['country_code'] == '':
            ip_location.country = None
        else:
            ip_location.country = content['country_code']

        if content['region'] == '':
            ip_location.region = None
        else:
            ip_location.region = content['region']

```

```
53          if content['city'] == '':
54              ip_location.city = None
55          else:
56              ip_location.city = content['city']
57
58          if content['latitude'] != '-' and content['longitude'] != '-':
59              ip_location.latitude = float(content['latitude'])
60              ip_location.longitude = float(content['longitude'])
61          else:
62              ip_location.latitude = None
63              ip_location.longitude = None
64
65          return ip_location
```

Výpis 4.15: Ukázka implementace třídy `Ipdata`

## 4.5 Vystavení aplikace

Jedním ze stěžejních bodů mé aplikace by mělo být, aby ji mohly použít i subjekty mimo moji fakultu, resp. mimo moji univerzitu. Abych usnadnil práci s mojí aplikací, resp. s celým balíčkem `ip2geotools`, rozhodl jsem se celou práci zveřejnit pod licencí MIT[38], což je svobodná licence a software pod touto licencí je možné použít zdarma pro komerční i nekomerční použití, v proprietárním software i v open-source sféře.

Celé znění licence je následující:

*MIT License*

*Copyright (c) 2017 Tomas Caha, tomas-net at seznam dot cz*

*Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:*

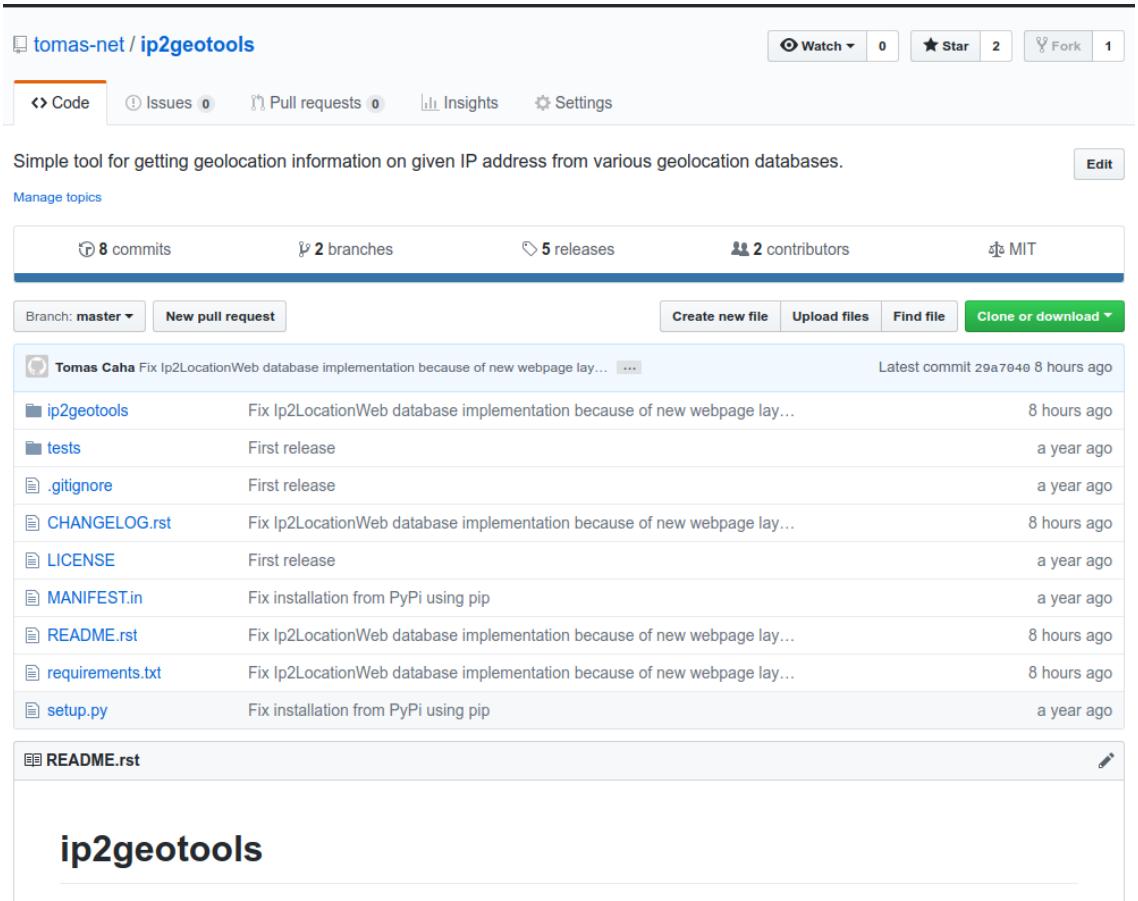
*The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.*

*THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.*

### 4.5.1 Způsob vystavení v repozitáři na GitHub

GitHub je webová platforma pro hostování repozitářů založených na verzovacím nástroji Git využívaného při vývoji počítačového softwaru. Služba umožňuje hostovat veřejně přístupné i soukromé (veřejně nepřístupné) repozitáře zcela zdarma. Projekty hostované na této službě mohou využívat i dalších možností této služby, jako jsou např. rozhraní pro hlášení chyb, procházení historie daného repozitáře, zobrazování nápověd a dokumentací[39]. Služba GitHub se de facto stala standardem pro zveřejňování zdrojových kódů open-source aplikací, a to je jeden z důvodů, proč jsem svoji aplikaci `ip2geotools` na této službě zveřejnil.

Na obr. 4.7 můžeme vidět, jak vypadá repozitář s kompletním zdrojovým kódem mé aplikace zveřejněný na GitHubu.

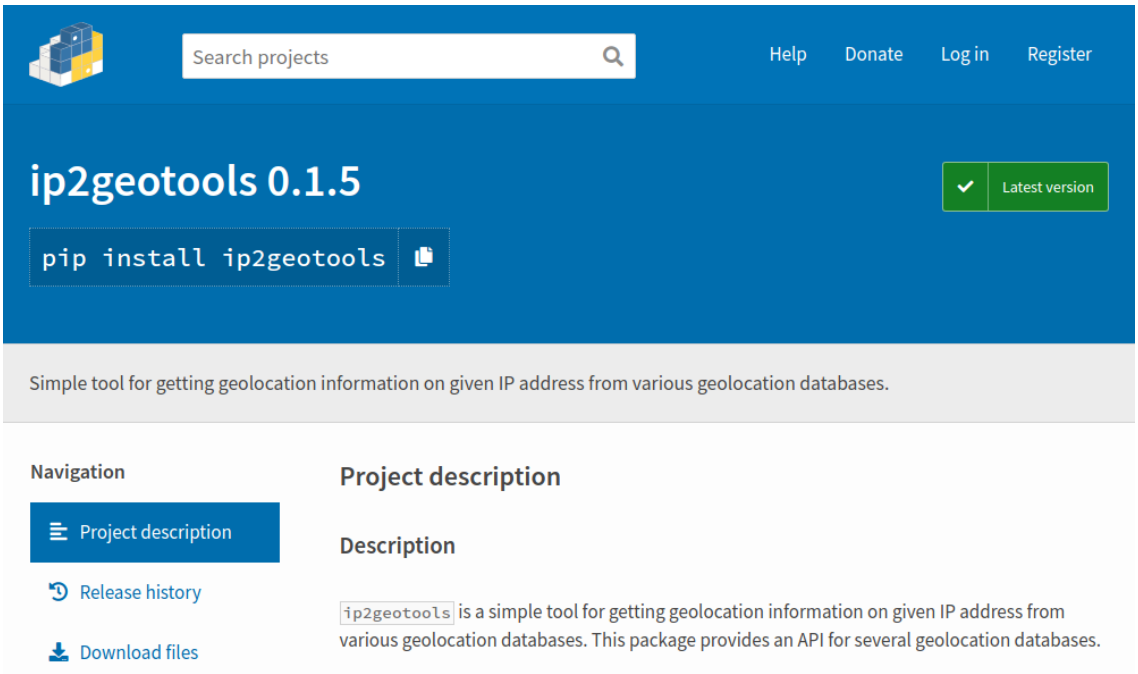



Obr. 4.7: Ukázka repozitáře na GitHubu s kompletním zdrojovým kódem mého balíčku `ip2geotools`

### 4.5.2 Způsob vystavení v archivu PyPi

PyPi, celým názvem „Python Package Index“, je veřejný seznam balíčků open-source knihoven pro Python. V tomto archivu jsou uloženy nejen zdrojové kódy (a jejich různé verze), ale také další metadata – autor, licence, kategorie a nápověda. Tento archiv je možné prohledávat podle klíčových slov nebo filtrovat dle kategorií[40].

Velkou výhodou tohoto centrálního archivu je existence správce balíčků `pip`, který je přímo součástí instalace prostředí Python (od verze 3.4, pro nižší verzi je nutné nainstalovat zvlášť). Díky tomu mohou vývojáři pracující v Pythonu efektivně a jednoduše využívat balíčky vytvořené jinými vývojáři a zrychlit tak svoji práci. Z tohoto důvodu jsem se rozhodl můj balíček `ip2geotools` v tomto centrálním archivu zveřejnit a umožnit tak jednoduchou a rychlou instalaci. Ukázku stránky, kde je můj balíček `ip2geotools` zveřejněn, lze vidět na obr. 4.8




Obr. 4.8: Ukázka stránky s mým balíčkem `ip2geotools` ve veřejném seznamu balíčků open-source knihoven pro Python – PyPi

Balíček `ip2geotools` s celou aplikací včetně všech závislostí je pak možné nainstalovat v prostředí Python 3, resp. Python 3.4 jediným příkazem:

```
1 $ pip install ip2geotools
```

Obdobně je možné balíček odinstalovat:

```
$ pip uninstall ip2geotools
```

Zveřejnění vlastního balíčku v centrálním archivu neprobíhá automaticky. Je nutné vytvořit soubor `setup.py`, který musí obsahovat v přesně daném formátu určité údaje, jedná se především o:

- název balíčku, který nesmí být již zabraný,
- verzi balíčku,
- krátký popis,
- detailní popis,
- autor a jeho email,
- místo, kde lze nalézt zdrojový kód
  (obvykle URL adresa repozitáře na GitHubu),
- místo, odkud lze stáhnout danou verzi
  (obvykle URL adresa vedoucí do archivu vydání na GitHubu),
- licence, resp. text licence,
- kategorie, do kterých se má balíček zařadit pro lepší filtrování na stránkách pypi.org,
- seznam závislostí, které jsou potřeba k bezproblémovému běhu,
- seznam souborů, ze kterých se má sestavit výsledný „instalační soubor“.

V souboru `setup.py` je možno používat běžné funkce, a proto jsem ho vyrobil tak, že se sestaví víceméně dynamicky těsně před zveřejňováním nové verze do centrálního archivu – předejde se tak např. chybně zapsané verzi, potřebných závislostí apod. Jak takový soubor vypadá, lze vidět níže na výpisu 4.16:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import io
import os
from setuptools import setup, find_packages
import ip2geotools


here = os.path.abspath(os.path.dirname(__file__))

with io.open(os.path.join(here, 'README.rst'), encoding='utf-8') as f:
    readme = '\n' + f.read()

with io.open(os.path.join(here, 'LICENSE'), encoding='utf-8') as f:
    license = '\n' + f.read()

with io.open(os.path.join(here, 'requirements.txt'), encoding='utf-8') as f:
    requirements = [i.strip() for i in f.readlines()]

setup(
```

```
22      name='ip2geotools',
23      version=ip2geotools.__version__,
24      description=ip2geotools.__description__,
25      long_description=readme,
26      author=ip2geotools.__author__,
27      author_email=ip2geotools.__author_email__,
28      url=ip2geotools.__url__,
29      download_url=ip2geotools.__url__ + '/archive/' + ip2geotools.__version__ + '.
            tar.gz',
30      packages=find_packages(exclude=['docs', 'tests', 'tests.*']),
31      package_data={'': ['LICENSE']},
32      package_dir={'ip2geotools': 'ip2geotools'},
33      install_requires=requirements,
34      include_package_data=True,
35      test_suite="tests",
36      license=ip2geotools.__license__,
37      classifiers=[
38          'Development Status :: 5 - Production/Stable',
39          'Environment :: Console',
40          'Intended Audience :: Developers',
41          'Intended Audience :: System Administrators',
42          'Intended Audience :: Telecommunications Industry',
43          'License :: OSI Approved :: MIT License',
44          'Natural Language :: English',
45          'Operating System :: POSIX :: Linux',
46          'Programming Language :: Python',
47          'Programming Language :: Python :: 3.3',
48          'Programming Language :: Python :: 3.4',
49          'Programming Language :: Python :: 3.5',
50          'Programming Language :: Python :: 3.6',
51          'Programming Language :: Python :: 3 :: Only',
52          'Programming Language :: Python :: Implementation :: CPython',
53          'Programming Language :: Python :: Implementation :: PyPy',
54          'Topic :: Internet',
55          'Topic :: Scientific/Engineering :: Information Analysis',
56          'Topic :: Utilities',
57      ],
58      entry_points={
59          'console_scripts': ['ip2geotools=ip2geotools.cli:execute_from_command_line'
                ],
60      },
61  )
```

Výpis 4.16: Soubor `setup.py` obsahující informace o balíčku `ip2geotools`

## 4.6 Možné způsoby použití aplikace

Vytvořenou aplikaci lze v podstatě použít dvěma způsoby, a to buď přímo spuštěním z příkazového řádku nebo začleněním do jiného programu vytvořeného v programovacím jazyce Python. V obou případech je nutné aplikaci nejdříve nainstalovat. Postup instalace z centrálního archivu balíčků již byl popsán výše. Instalaci lze provést jednoduše pomocí:

```
$ pip install ip2geotools
```

### 4.6.1 Jako samostatný program

Až do tohoto momentu vše výše popsané tvoří pouze sadu nástrojů pro geolokaci IP adres. Tato sada nástrojů ale sama o sobě netvoří žádný spustitelný program, a proto jsem ji doplnil o jednoduchý jednoúčelový program, který je spustitelný v příkazové řádce. Samotné srdce programu je součástí modulu `ip2geotools.cli`. V okamžiku, kdy máme balíček `ip2geotools` nainstalovaný, můžeme spustit jednoduchý geolokační program příkazem:

```
$ ip2geotools
```

Použití programu je následující:

```
ip2geotools [-h] -d {dbipcity,hostip,freegeoip,ipstack,
    maxmindgeolite2city,ip2location,dbipweb,maxmindgeoip2city,
    ip2locationweb,neustarweb,geobytescitydetails,
    skyhookcontextacceleratorip,ipinfo,eurek,ipdata}
                [--api_key API_KEY] [--db_path DB_PATH] [-u
                    USERNAME]
                [-p PASSWORD] [-f {json,xml,csv-space,csv-tab,
                    inline}] [-v]
                IP_ADDRESS
```

kde:

- `ip2geotools` je název spustitelného skriptu,
- `IP_ADDRESS` je IP adresa, kterou chceme lokalizovat,
- `-h`, `--help` je parametr pro zobrazení nápovědy a ukončení programu,
- `-d dbipcity,hostip,...,eurek` je název geolokační databáze, kterou chceme použít (nezáleží na velikosti písmen),
- `--api_key API_KEY` je parametr pro zadání API klíče pro přístup do databáze (pokud je potřeba),
- `--db_path DB_PATH` je parametr pro určení cesty k souboru s geolokačními daty (pokud je potřeba),
- `-u USERNAME`, `--username USERNAME` je parametr pro zadání uživatelského jména pro přístup do databáze (pokud je potřeba)
- `-p PASSWORD`, `--password PASSWORD` je parametr pro zadání hesla pro přístup do databáze (pokud je potřeba),

- `-f json,xml,csv-space,csv-tab,inline`,
  `--format json,xml,csv-space,csv-tab,inline` je parametr určující formát, v jakém bude výsledek geolokace zobrazen,
- `-v`, `--version` je parametr pro zobrazení verze této aplikace a ukončení programu.

Níže jsou příklady použití:

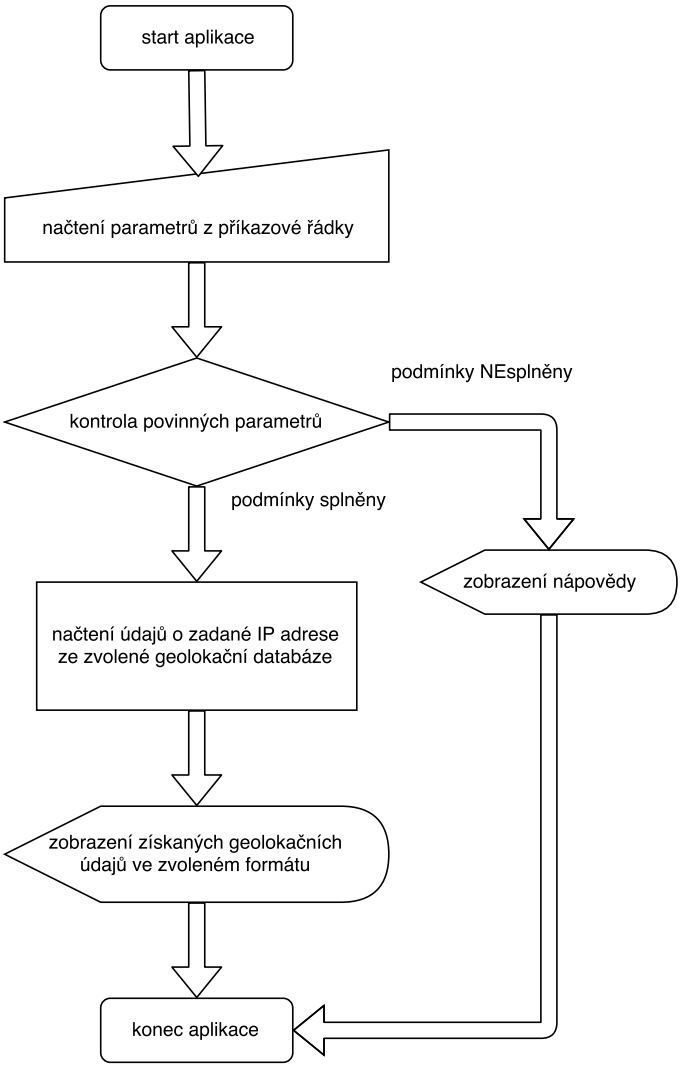
```
1  $ ip2geotools 147.229.2.90 -d dbipcity -f json
2  {"longitude": 16.6182105, "country": "CZ", "ip_address": "
       147.229.2.90", "city": "Brno (Brno střed)", "latitude":
       49.1926824, "region": "South Moravian"}
3
4  $ ip2geotools 147.229.2.90 -d ip2locationweb -f inline
5  147.229.2.90
6  Brno
7  Jihomoravsky kraj
8  CZ
9  49.19522
10 16.60796
```

Na obr. 4.9 je zjednodušeně znázorněn vývojový diagram tohoto programu. Při startu programu se provede načtení parametrů z příkazového řádku a následně se provede jejich kontrola. Jestliže povinné parametry nejsou dostupné, je vypsána nápověda. V případě, že jsou parametry v pořádku, vybere se implementace konkrétní geolokační databáze z modulu `ip2geotools.databases.commercial` nebo `ip2geotools.databases.noncommercial` a načtou se z ní geolokační údaje. Pokud by nastala nějaká výjimka, která pochází z mé vlastní sady výjimek z modulu `ip2geotools.errors`, je zpracována a zobrazena uživateli ve zvoleném formátu. Jestliže získání údajů z geolokační databáze proběhlo bez problémů, tak vzhledem k tomu, že všechny databáze implementují mé vlastní rozhraní `IGeoIpDatabase` z modulu `ip2geotools.databases.interfaces`, máme jistotu, že údaje z databáze máme ve formě objektu třídy `IpLocation` z modulu `ip2geotools.models`.

Tento jednoduchý program zajišťuje základní použitelnost celé mé aplikace a dále demonstruje, že veškeré mnou vytvořené moduly do sebe zapadají jako skládačka, logicky na sebe navazují a vzájemně spolupracují.

### 4.6.2 Jako součást jiných programů

Druhým způsobem, jak lze využít mojí aplikace, je začlenit ji do jiného programu, resp. využít ji při tvorbě nějakého nového programu v jazyce Python. Jakmile




Obr. 4.9: Vývojový diagram konzolové části aplikace `ip2geotools`

máme balíček `ip2geotools` nainstalovaný, můžeme jednotlivé části aplikace začlenit do svého programu[41]. Na výpisu 4.17 jsem uvedl jednoduchý příklad použití, kde v konzolovém rozhraní Python naimportuji z modulu `ip2geotools.databases.noncommercial` třídu `DbIpCity`, provedu dotaz na IP adresu za využití volného API klíče `free`, vypíšu vybrané údaje, resp. nechám vypsat všechny získané údaje ve vybraných formátech.

```
>>> from ip2geotools.databases.noncommercial import DbIpCity
>>> response = DbIpCity.get('147.229.2.90', api_key='free')
>>> response.ip_address
'147.229.2.90'
>>> response.city
'Brno (Brno střed)'
>>> response.region
'South Moravian'
>>> response.country
```

```
'CZ'
>>> response.latitude
49.1926824
>>> response.longitude
16.6182105
>>> response.to_json()
'{
  "ip_address": "147.229.2.90",
  "city": "Brno (Brno střed)",
  "region": "South Moravian",
  "country": "CZ",
  "latitude": 49.1926824,
  "longitude": 16.6182105
}'
>>> response.to_xml()
'<?xml version="1.0" encoding="UTF-8" ?>
<ip_location>
  <ip_address>147.229.2.90</ip_address>
  <city>Brno (Brno střed)</city>
  <region>South Moravian</region>
  <country>CZ</country>
  <latitude>49.1926824</latitude>
  <longitude>16.6182105</longitude>
</ip_location>'
>>> response.to_csv(',')
'147.229.2.90,Brno (Brno střed),South Moravian,CZ,49.1926824,16.6182105'
```

Výpis 4.17: Ukázka použití balíčku `ip2geotools` v konzolovém rozhraní jazyka Python

Příklad výše demonstruje nejjednodušší možné začlenění mé aplikace do programu jiného. Další využití je demonstrováno v kapitole 5, kdy máme větší množství IP adres a potřebujeme k nim dohledat geografickou polohu. Aplikace `ip2geotools` sama o sobě takovou funkci nemá, nicméně poskytuje velmi solidní základ a nástroje, že není problém napsat zcela nový program, který takovou funkcionalitu poskytne.

# 5 Geografická analýza zdrojů kybernetických útoků

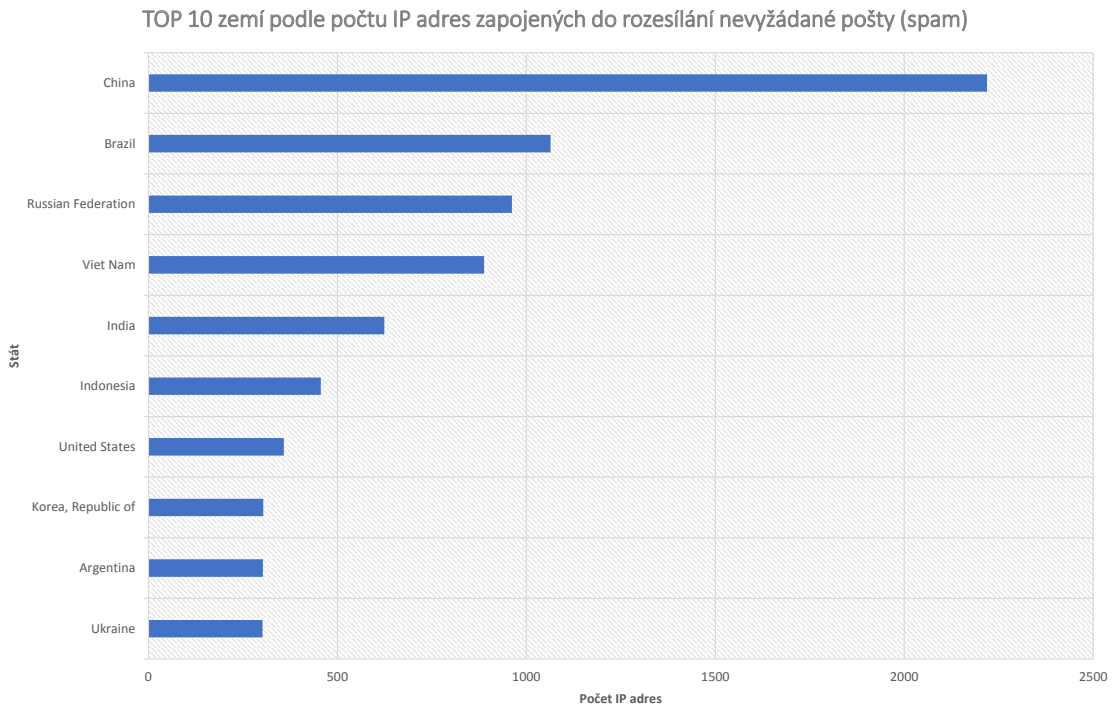
Dalším z cílů mé práce je analyzovat zdroje nevyžádané pošty a kybernetických útoků, a proto jsem společně s vedoucím práce vybral volně dostupné seznamy obsahující větší množství IP adres. Pro statistické účely jsme vybrali seznamy s alespoň 750 záznamy. Většina analyzovaných seznamů má ale počty záznamů v řádu tisíců nebo více.

Analýza se skládá ze dvou částí – z jednoduchého skriptu `ipanalysis.py` napsaného v jazyce Python využívajícího moji aplikaci `ip2geotools`, který má za cíl provést hromadnou geolokaci IP adres uvedených na vybraných seznamech, získané výsledky uložit ve formátu CSV pro další zpracování v Excelu a ze získaných výsledků vytvořit tzv. „heat mapu“. Ke geografické lokalizaci IP adres byla použita volně dostupná geolokační databáze MaxMind ve verzi GeoLite2City, která byla stažena 26. března 2019 v 10:30 z `https://geolite.maxmind.com/download/geoip/database/GeoLite2-City.tar.gz`.

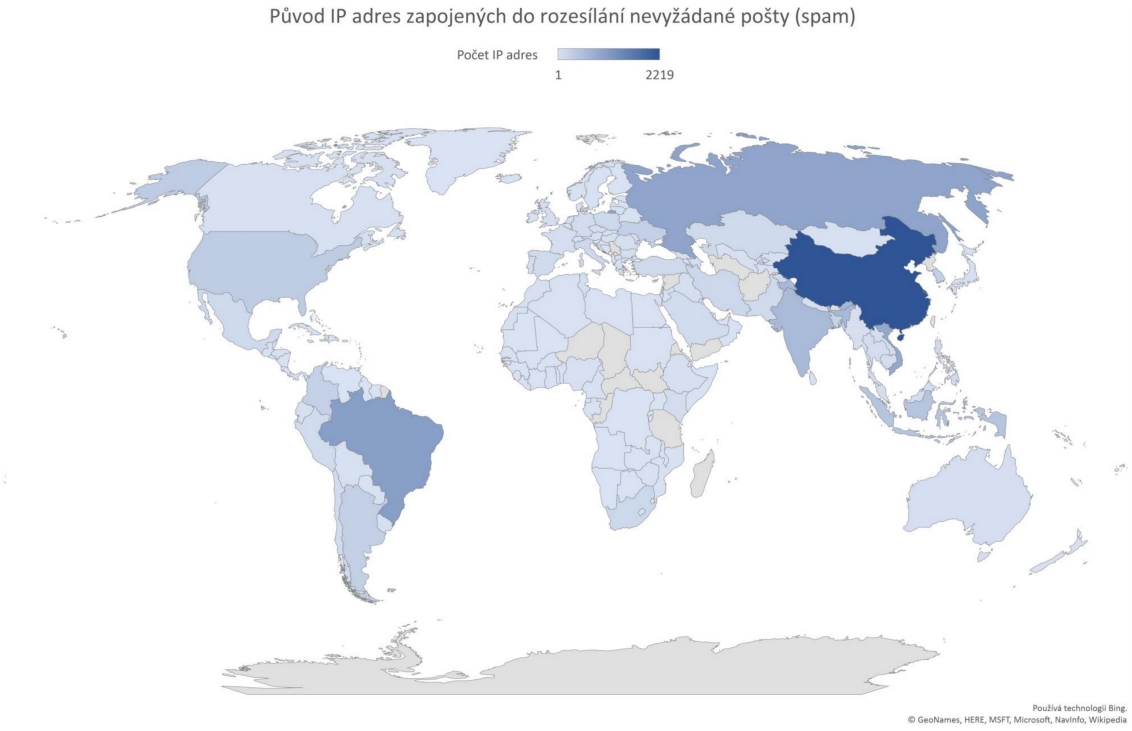
## 5.1 Nevyžádaná pošta (spam)

K analýze zdrojů nevyžádané pošty jsem vybral seznam IP adres s názvem NiX Spam, který obsahuje IP adresy e-mailových serverů, které za poslední 1 hodinu odesílaly e-maily klasifikované jako spam. IP adresy jsou ze seznamu odebírány po 12 hodinách od posledního zaznamenaného odeslaného nevyžádaného e-mailu[42].

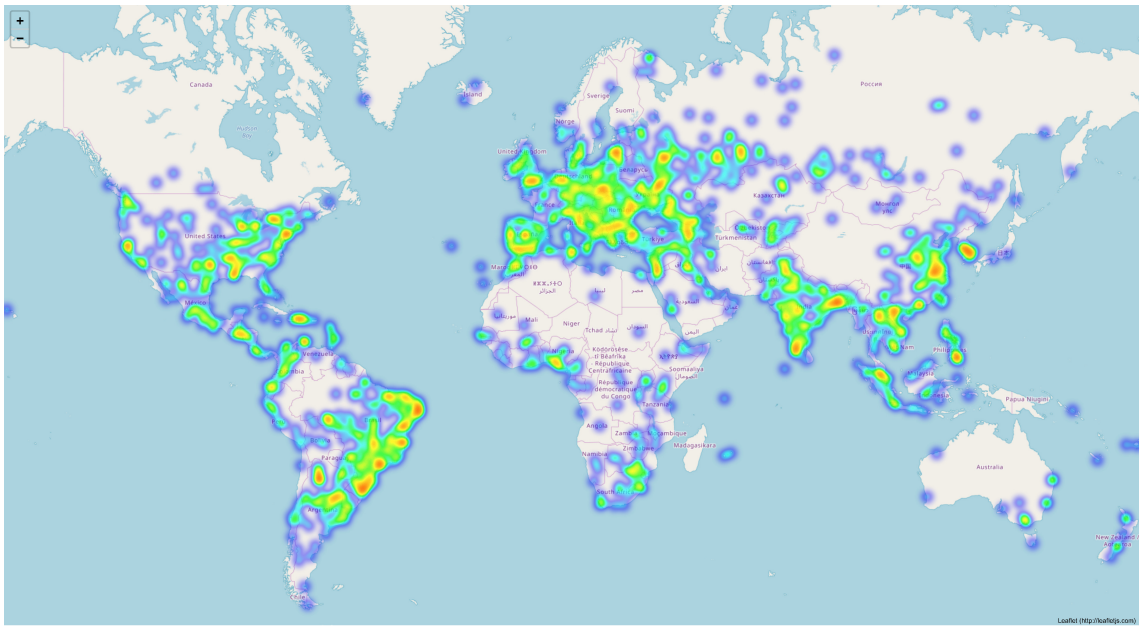
Seznam IP adres byl stažen 26. března 2019 v 10:30 z portálu FireHOL IP Lists[43], který obsahoval 12 440 záznamů. Seznam byl zpracován mým skriptem popsaným výše s následujícími výsledky. Na obr. 5.1 je graf, na kterém můžeme vidět, že největším odesílatelem nevyžádané pošty je Čína. Dalšími zeměmi jsou Brazílie, Rusko a Vietnam, které mají velmi podobný podíl. Obr. 5.2 zobrazuje stejné údaje na světové mapě. Podrobnější data nám poskytne pohled na tzv. „heat mapu“ na obrázku 5.3, na které můžeme vidět, že nejvíce nevyžádané pošty je odesíláno z IP adres lokalizovaných v Šanghaji.




Obr. 5.1: TOP 10 zemí podle počtu IP adres zapojených do rozesílání nevyžádané pošty (spam)




Obr. 5.2: Původ IP adres zapojených do rozesílání nevyžádané pošty (spam)

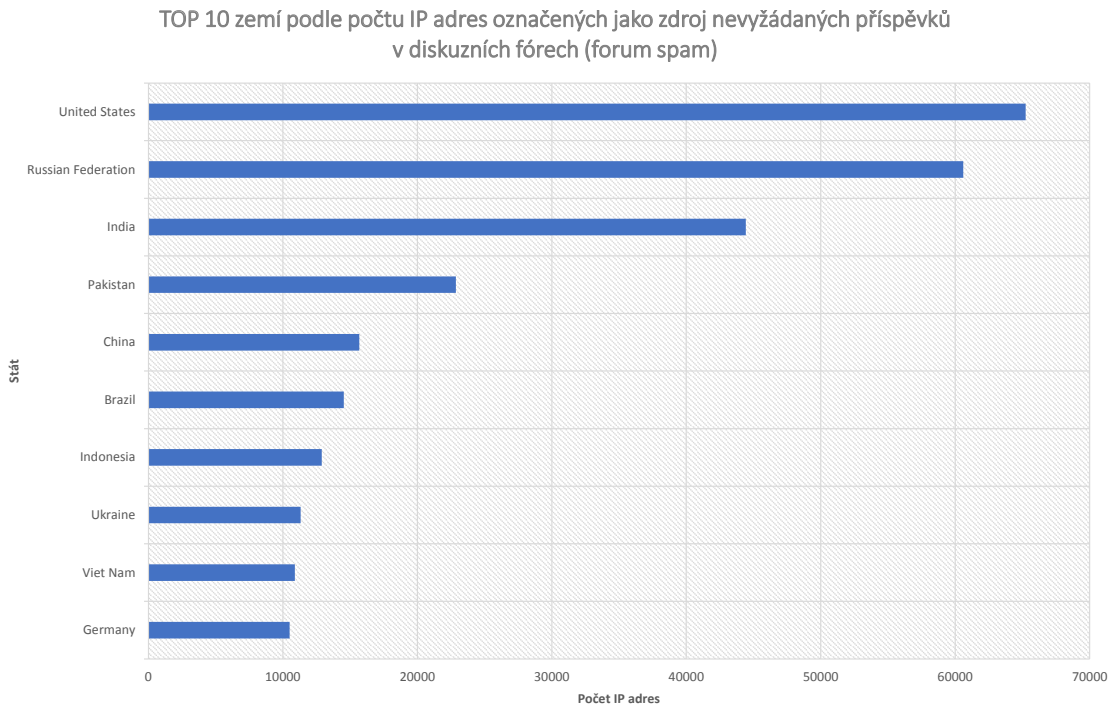


Obr. 5.3: Heat mapa původu IP adres zapojených do rozesílání nevyžádané pošty (spam)

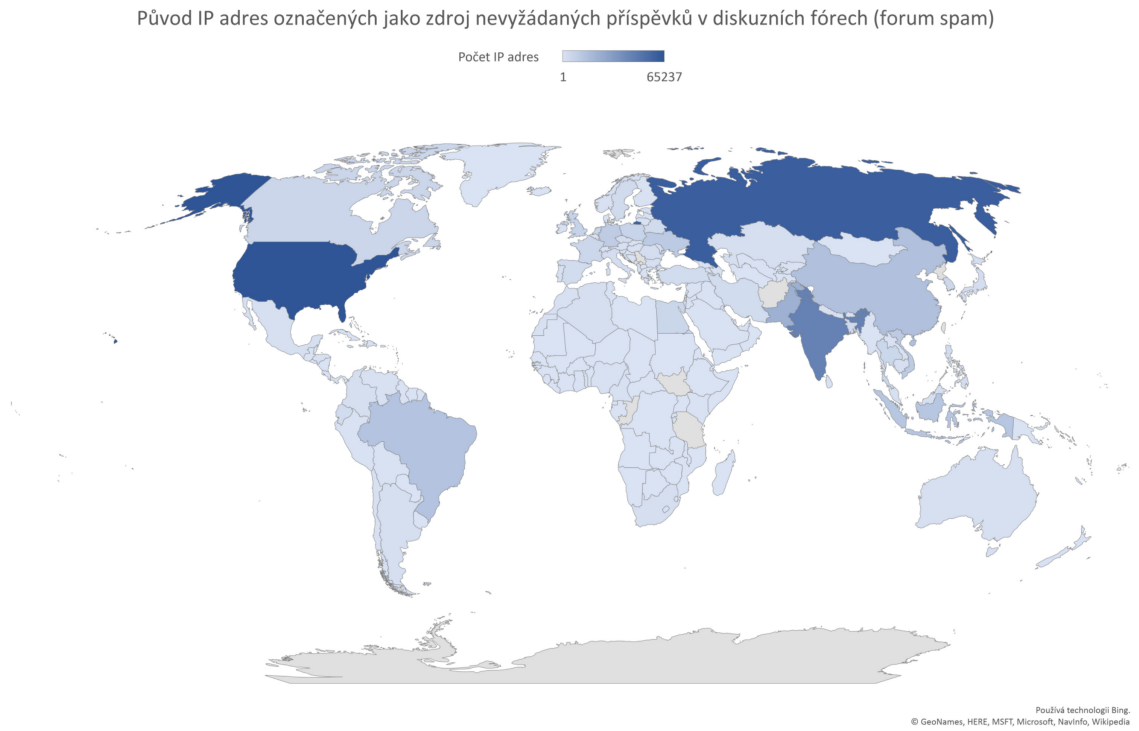
## 5.2 Nevyžádané příspěvky v diskuzních fórech (forum spam)

Pracoval jsem se s daty, které poskytuje služba Stop Forum Spam. Tato služba sestavuje několik typů seznamů, které mají za cíl poskytnout jednoduchou cestu k zablokování uživatelů, kteří vkládají do diskuzních fór nebo do komentářů pod různé články nevhodné příspěvky. Za nevhodný příspěvek může být považován například příspěvek obsahující pouze odkazy, jejichž cílem je reklama. Dalším příkladem mohou být příspěvky s náhodně generovaným textem na určité téma, které je proloženo URL adresami odkazujícími na nevhodné weby[44].

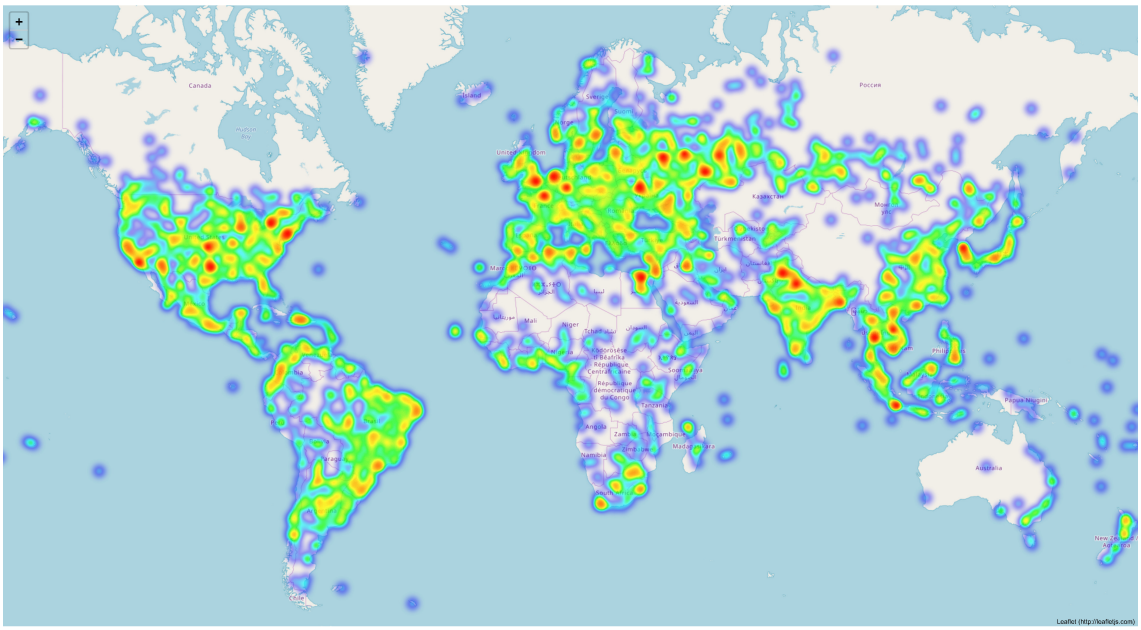
K analýze jsem vybral seznam IP adres, které byly nahlášeny alespoň jedenkrát za poslední 1 rok. Seznam byl stažen 26. března 2019 v 10:30 z portálu FireHOL IP Lists[45] a obsahoval 410 925 záznamů. Výsledky zpracování lze vidět na obr. 5.4, na kterém je graf ukazující, že dvěma největšími původci nevyžádaných příspěvků v diskuzních fórech jsou Spojené státy americké a Rusko. Na třetím místě s odstupem přibližně 15 000 IP adres je Indie. I další země na grafu mají mezi 10 000 a 20 000 IP adresami. Z obr. 5.5, který zobrazuje údaje na světové mapě, je patrné, že zbývající státy jsou si počtem IP adres rovnocenné. Z heat mapy na obrázku 5.6 je čitelné, že státy v Africe nejsou do těchto činností významně zapojené.




Obr. 5.4: TOP 10 zemí podle počtu IP adres označených jako zdroj nevyžádaných příspěvků v diskuzních fórech (forum spam)




Obr. 5.5: Původ IP adres označených jako zdroj nevyžádaných příspěvků v diskuzních fórech (forum spam)

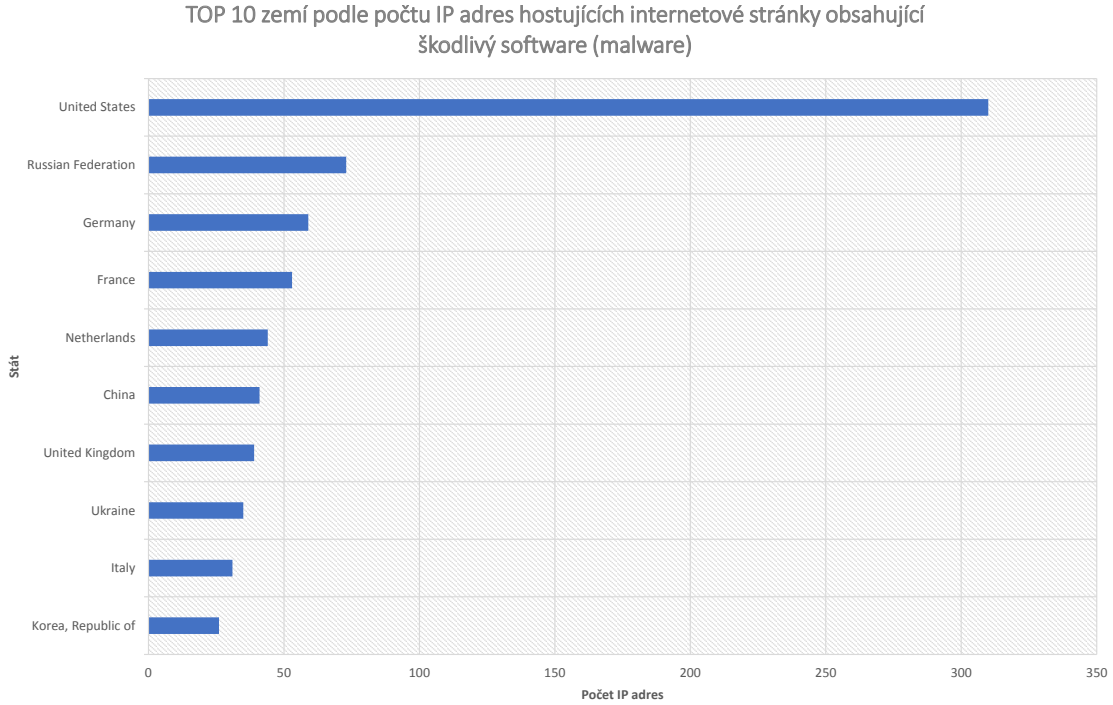


Obr. 5.6: Heat mapa původu IP adres označených jako zdroj nevyžádaných příspěvků v diskuzních fórech (forum spam)

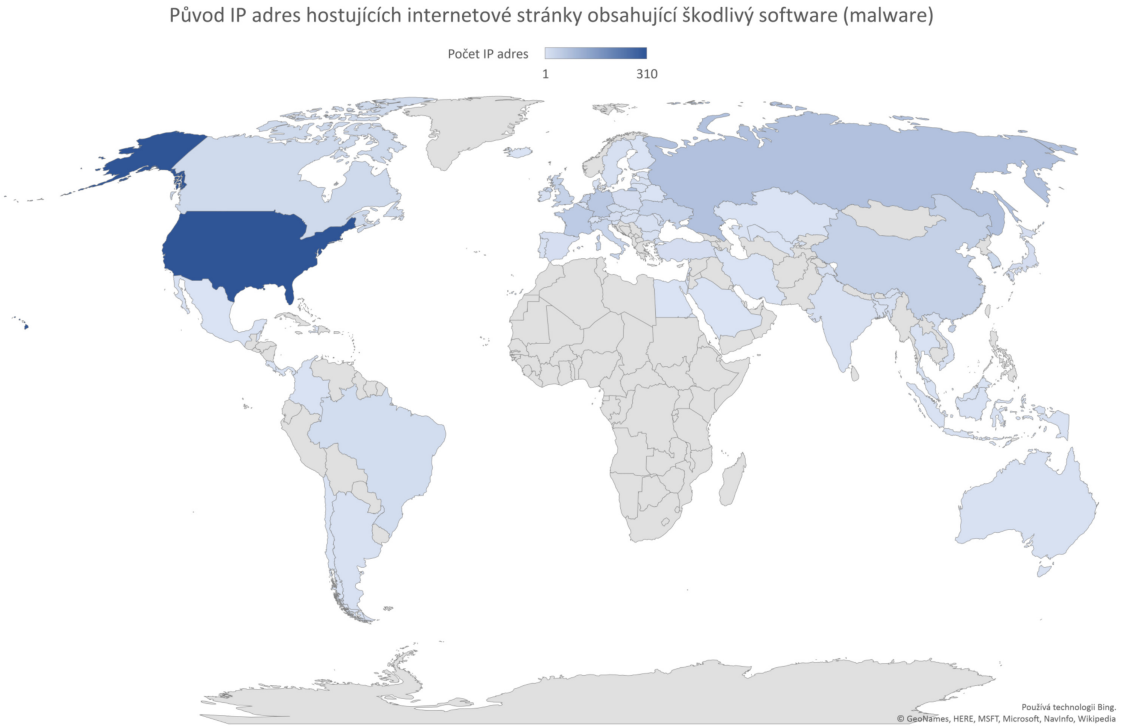
## 5.3 Škodlivý software (malware)

K analýze jsem vybral data, která poskytuje nekomerční komunitní projekt Malware Domain List sdružující nadšence a bezpečnostní analytiky. Zástupce projektu tvrdí, že všechny domény, resp. IP adresy uvedené na jejich seznamu by měly být považované za vysoce nebezpečné. Hlášení nových výskytů probíhá pomocí diskuzního fóra tohoto projektu. Každý záznam obsahuje i krátký popis, proč byl na seznam nebezpečných webů přidán[46][47].

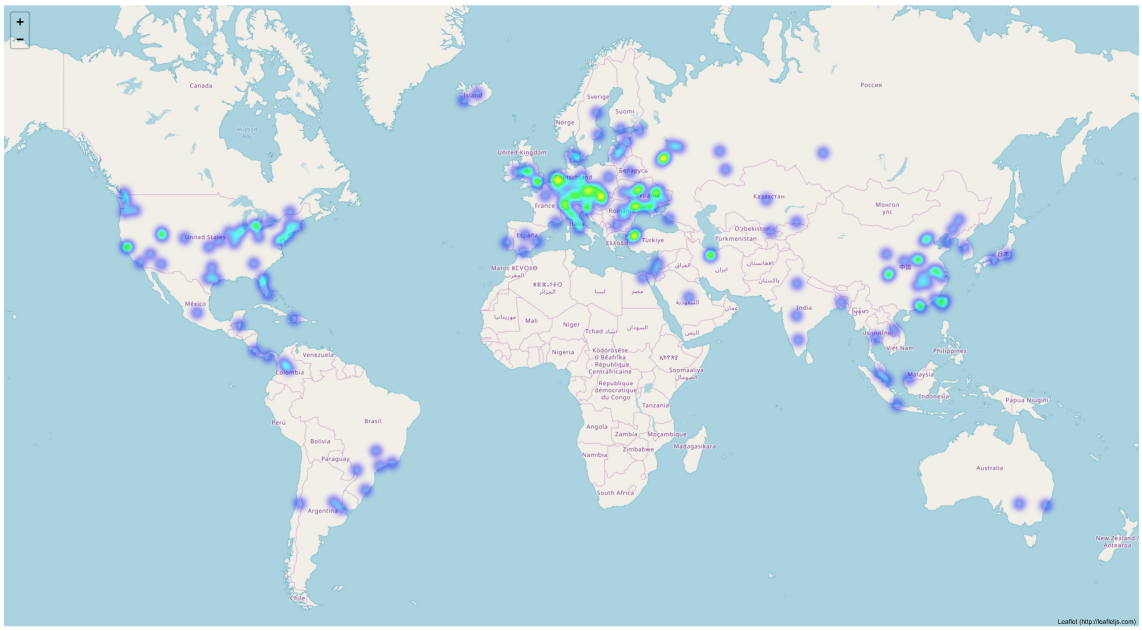
Seznam IP adres byl stažen 26. března 2019 v 10:30 z portálu FireHOL IP Lists[48], který obsahoval 997 záznamů. Na obr. 5.7 je graf, který nám ukazuje, že zemí s největším počtem IP adres, na kterých je hostovaný škodlivý software, jsou Spojené státy americké, což se mi zdá být logické – webové stránky hostující škodlivý software budou zřejmě v pozici obětí. Záškodníci budou cílit svůj škodlivý software na bohatší země. Ostatní země zastoupené v TOP 10 obsahují přibližně 30 až 60 IP adres. Obr. 5.8 zobrazuje obdobné údaje na světové mapě, na které můžeme vidět, že africké státy nejsou vůbec postiženy. Heat mapa na obrázku 5.9 výše popsané údaje mírně zkresluje, což je dáno způsobem seskupování výskytů kolem datových center v Evropě – Německo, Francie, Nizozemí, Velká Británie, Itálie, Česká republika.




Obr. 5.7: TOP 10 zemí podle počtu IP adres hostujících internetové stránky obsahující škodlivý software (malware)




Obr. 5.8: Původ IP adres hostujících internetové stránky obsahující škodlivý software (malware)

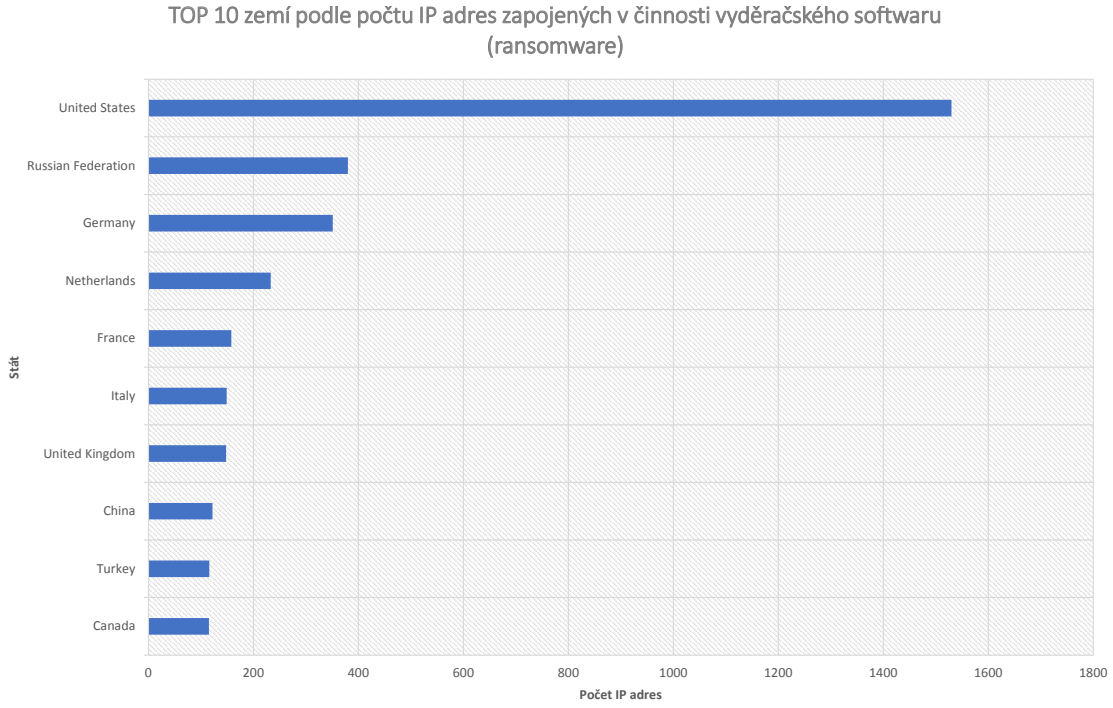



Obr. 5.9: Heat mapa původu IP adres hostujících internetové stránky obsahující škodlivý software (malware)

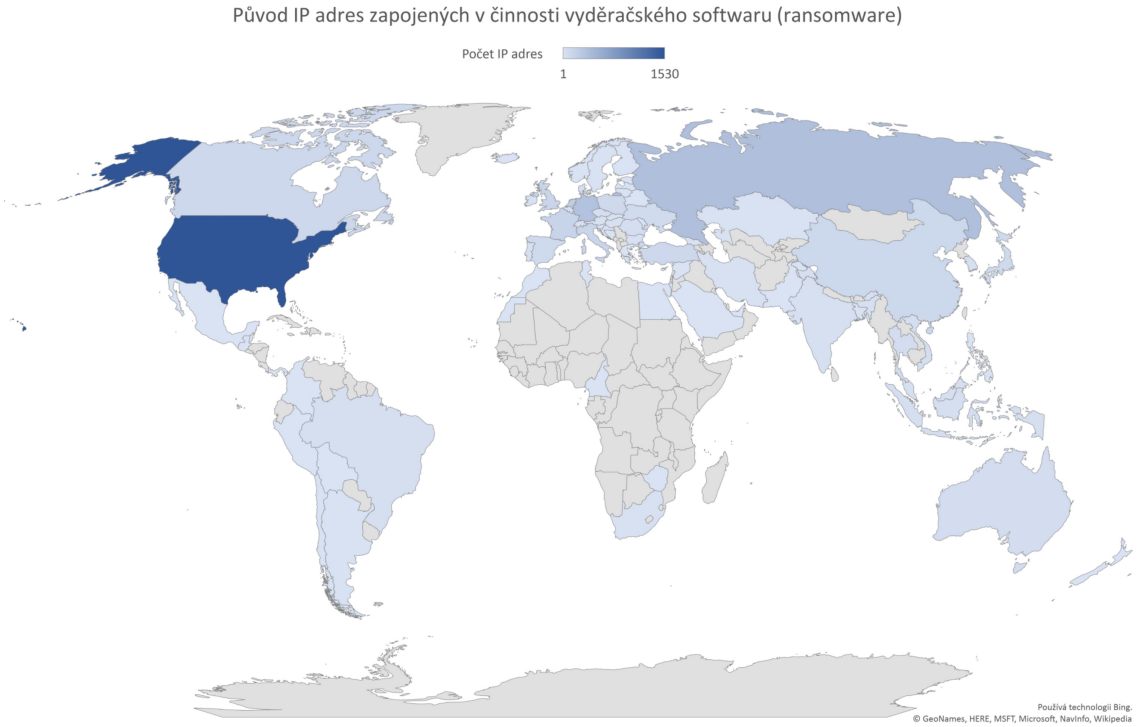
## 5.4 Vyděračský software (ransomware)

Vybral jsem si službu Ransomware Tracker sledující a evidující servery, které jsou nějakým způsobem zapojeny ve vyděračském software. Jedná se zejména o webové stránky, které slouží k distribuci takového softwaru, dále pak servery zprostředkovávající platby výkupného a v neposlední řadě také ostatní servery podílející se na ovládání vyděračského softwaru[49].

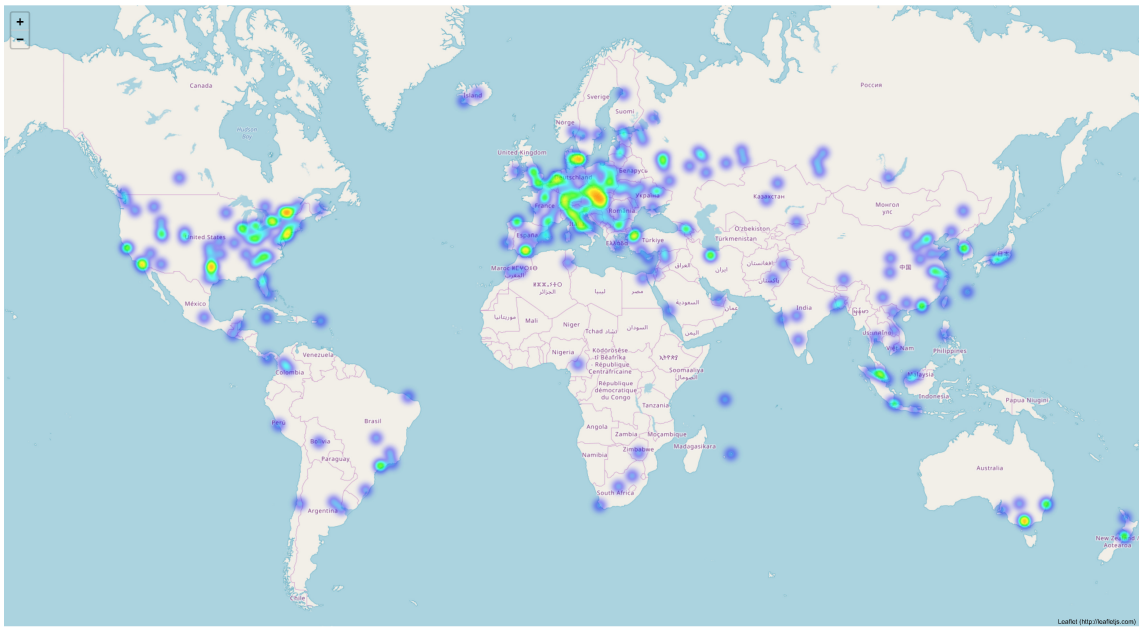
Seznam IP adres byl stažen 26. března 2019 v 10:30 z portálu FireHOL IP Lists[50] a obsahoval 4 758 záznamů. Výsledky lze vidět na obr. 5.10, kde je graf, na kterém můžeme vidět, že zemí s největším množstvím IP adres zapojených ve vyděračském softwaru jsou Spojené státy americké. S velkým odstupem se umístilo Rusko a Německo. S odstupem se umístily další evropské i mimo evropské země. Podle mého názoru jsou uvedené země obětmi a útočníci si servery (např. virtuální s IP adresami těchto zemí) jednoduše pronajímají. Obr. 5.11 zobrazující údaje na světové mapě nám dává informaci o tom, že státy v Africe nejsou na tomto seznamu zahrnuty. Z heat mapy na obrázku 5.11 je čitelné, že IP adresy byly lokalizované v blízkosti velkých datových center.




Obr. 5.10: TOP 10 zemí podle počtu IP adres zapojených v činnosti vyděračského softwaru (ransomware)




Obr. 5.11: Původ IP adres zapojených v činnosti vyděračského softwaru (ransomware)

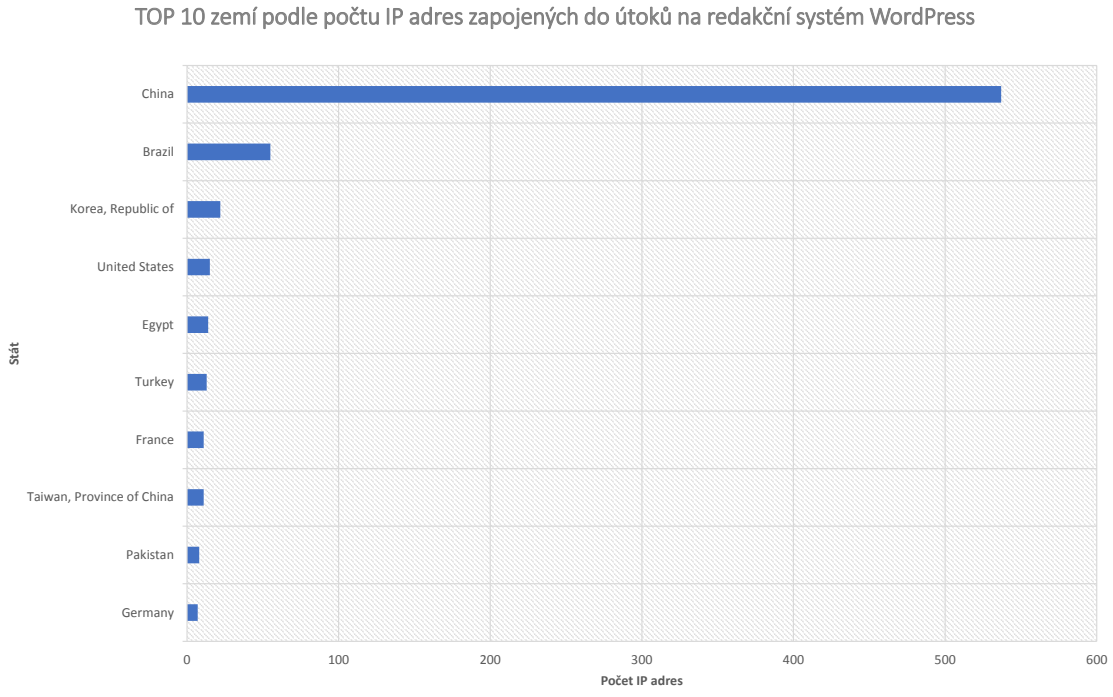



Obr. 5.12: Heat mapa původu IP adres zapojených v činnosti vyděračského softwaru (ransomware)

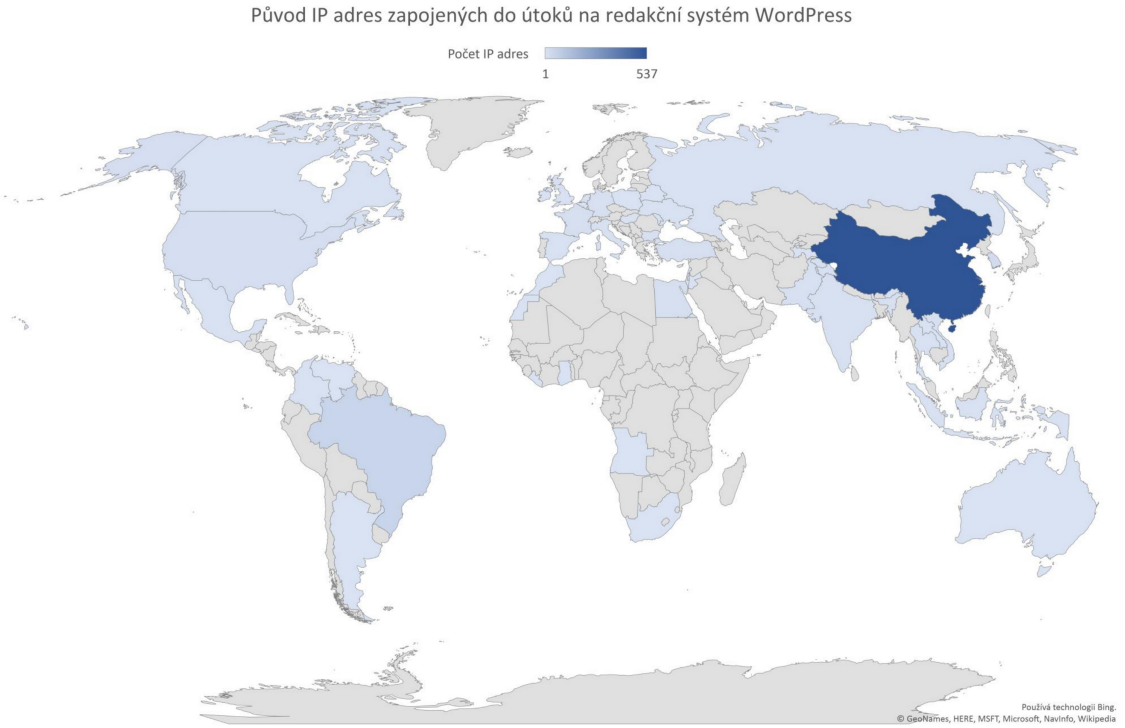
## 5.5 Útoky na redakční systém WordPress

K analýze jsem vybral data poskytované webem BadIPs.com. Jedná se o komunitní projekt vytvářející různé seznamy IP adres, které by bylo vhodné z určitých důvodů blokovat nebo omezovat jejich provoz[51].

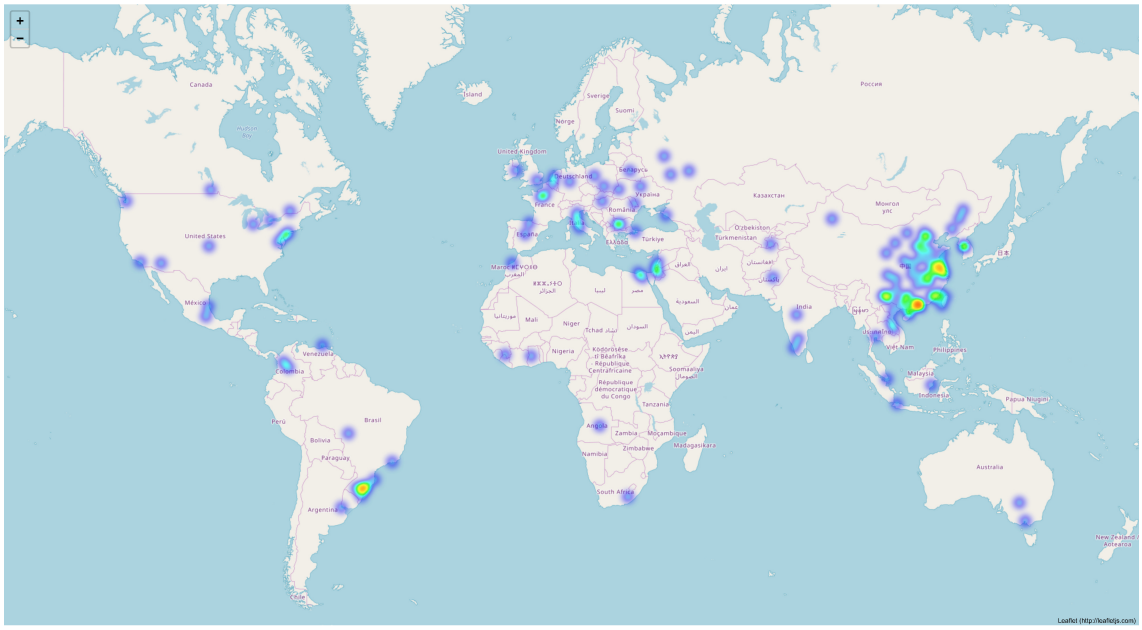
Pracoval jsem se seznamem IP adres se štítkem `wordpress`, který má skóre vyšší než 2 a není starší více než 30 dnů. Seznam byl stažen 26. března 2019 v 10:30 z portálu FireHOL IP Lists[52], který obsahoval 788 záznamů. Na obr. 5.13 je graf, který nám ukazuje, že drtivá většina útoků na redakční systém WordPress pochází dle IP adres z Číny. Na druhém místě se umístila Brazílie, ale její podíl je velmi malý. Zbývající země uvedené na stejném grafu jsou z pohledu počtu IP adres bezvýznamné. Velký podíl Číny bude pravděpodobně souviset s tím, že po průniku do redakčního systému je možné webové stránky infikovat a distribuovat přes ně např. škodlivý software, což by odpovídalo výsledkům výše. Z obrázků 5.14 i 5.15 vyplývá, že vyjma Číny a Brazílie jsou útoky na tento redakční systém z jiných zemí spíše výjimečné.




Obr. 5.13: TOP 10 zemí podle počtu IP adres zapojených do útoků na redakční systém WordPress



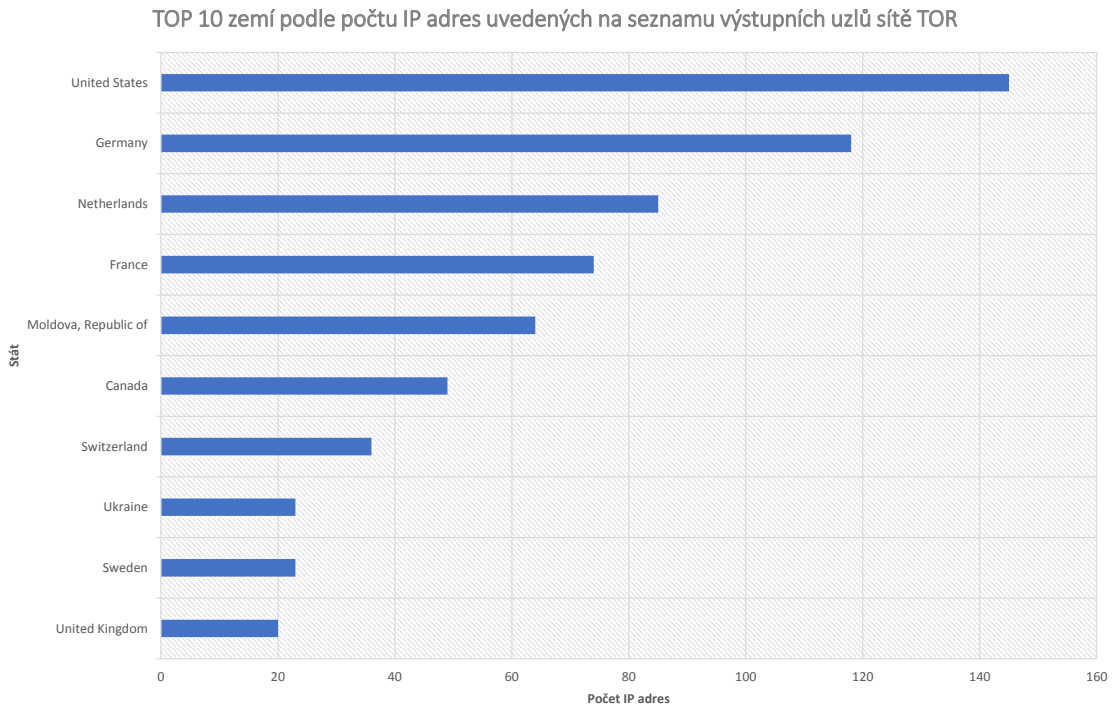

Obr. 5.14: Původ IP adres zapojených do útoků na redakční systém WordPress



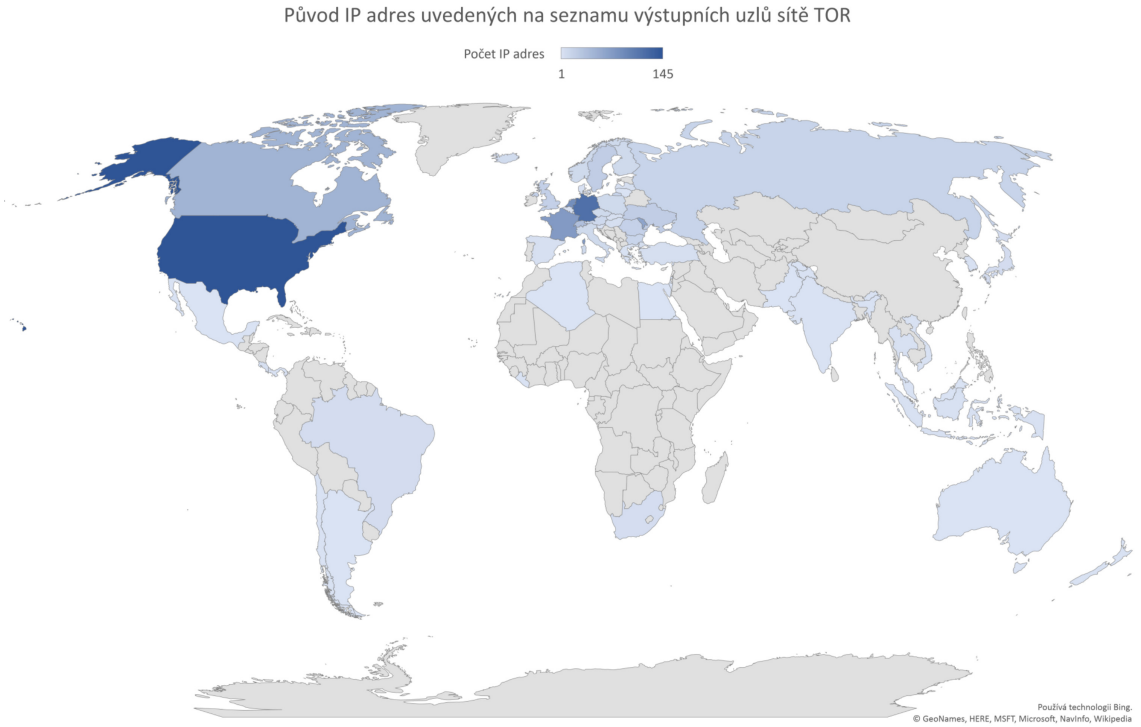

Obr. 5.15: Heat mapa původu IP adres zapojených do útoků na redakční systém WordPress

## 5.6 Výstupní uzly sítě Tor

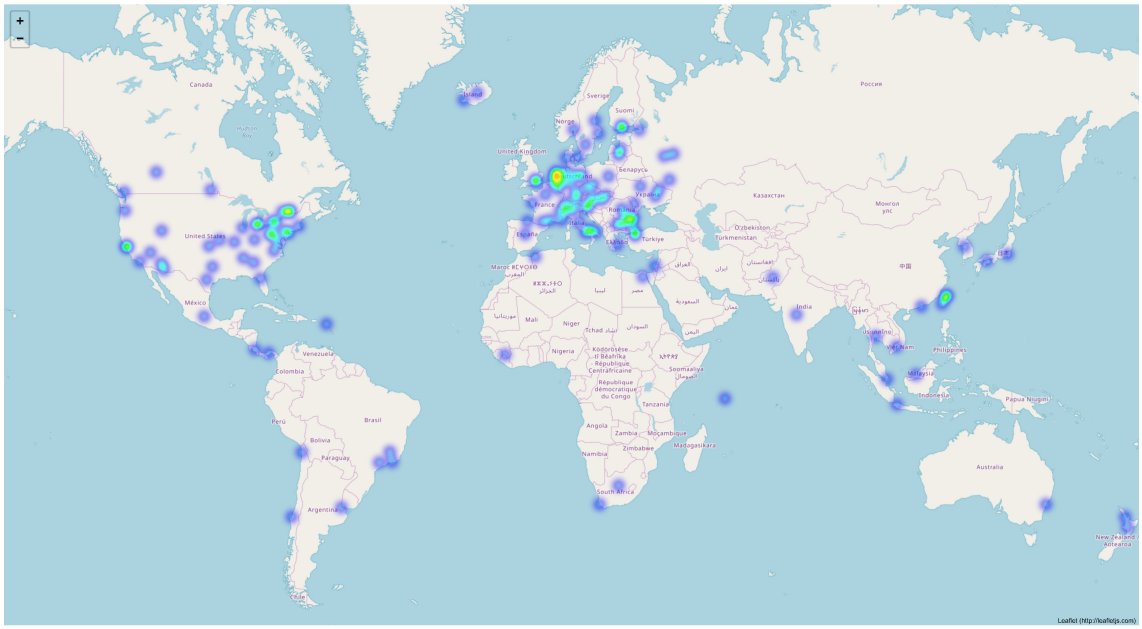
K analýze geografického umístění výstupních uzlů sítě Tor jsem vybral seznam IP adres projektu TorProject.org obsahující IP adresy všech dostupných výstupních uzlů. Seznam IP adres byl stažen 26. března 2019 v 10:30 z portálu FireHOL IP Lists[53], který obsahoval 884 záznamů. Na obr. 5.16 je graf, na kterém můžeme vidět, že zemí s největším množstvím výstupních uzlů jsou Spojené státy americké následované evropskými státy jako je Německo nebo Nizozemí. Přibližně polovina všech výstupních uzlů je umístěna v Evropě, což můžeme vidět na heat mapě na obrázku 5.18. Na obr. 5.17 s mapou světa můžeme vidět, že výstupní uzly se vůbec nenachází například v Číně, Mongolsku nebo ve spoustě afrických států.




Obr. 5.16: TOP 10 zemí podle počtu IP adres uvedených na seznam výstupních uzlů sítě Tor




Obr. 5.17: Původ IP adres uvedených na seznam výstupních uzlů sítě Tor




Obr. 5.18: Heat mapa původu IP adres uvedených na seznam výstupních uzlů sítě Tor

## 5.7 Srovnání výsledků s dalšími publikacemi

Podařilo se mi dohledat článek z roku 2004[54], ve kterém se zaměstnanci společnosti Microsoft zabývali nevyžádanou poštou. V článku mj. zjišťovali, ze kterých států je spam rozesílán. Na druhé straně článku je obrázek s mapou světa, kde jsou jako odesílatelé nevyžádané pošty zvýrazněny státy jako jsou Čína, Rusko, Indie, Indonésie, Pákistán a Brazílie, což odpovídá i mým poznatkům. V článku nicméně nejsou uvedeny žádné bližší údaje ani pořadí těchto států.

Publikace věnovaná nevyžádaným příspěvkům v diskuzních fórech z roku 2011[55] se mj. zaobírala tím, ze kterých zemí (na základě IP adresy) příspěvky pocházely. Na čtvrté straně článku je tabulka, která zobrazuje 10 zemí s největším počtem nevyžádaných příspěvků. Tabulka obsahuje země v tomto pořadí: Spojené státy americké, Rusko, Ukrajina, Čína, Německo, Indie, Izrael, Lotyšsko, Brazílie a Nizozemí. Tyto údaje přibližně odpovídají mým poznatkům, kdy Spojené státy americké a Rusko jsou dvěma největšími původci nevyžádaných příspěvků v diskuzních fórech a komentářích. V dalších zemích se v údajích liším – země jako Německo, Čína nebo Ukrajina mám v grafu taktéž uvedené ale s jiným podílem. Pak se zde objevují státy (Indie, Pákistán), které v mém grafu mají poměrně velký podíl, ale článek je nezmiňuje.

Poslední publikací, kterou se mi podařilo vyhledat, je publikace z roku 2018[56] věnovaná problematice vyděračského softwaru. Publikace popisuje společné znaky různých typů vyděračského softwaru a tvrdí, že vyděračský software pochází ze zemí jako jsou Čína, Rusko nebo Severní Korea, což částečně odpovídá i mým poznatkům – Rusko mám hned na 2. místě a Čínu na 8. místě. Severní Korea se ve zpracovaných výsledcích nenachází (nebo v zanedbatelné míře). Rozdíly budou způsobeny tím, že tato publikace zjišťovala původ tvůrce vyděračského softwaru, zatímco mnou zpracovávaný seznam se zabýval všemi zapojenými články celého procesu (distribuce vyděračského softwaru, platba, apod.).

# 6 Závěr

Na začátku této práce jsem se seznámil se způsoby, kterými je možné odhadovat geografickou pozici IP adres, a s jednotlivými volně dostupnými i komerčními geolokačními databázemi. Dále jsem popsal rozhraní jednotlivých geolokačních databází a množství informací, které poskytují. Seznámil jsem se s vybranými druhy kybernetických hrozeb.

Cílem této práce bylo vytvořit aplikaci pro odhad geografické polohy pro zadanou IP adresu. V práci jsem detailně rozebral návrh mé aplikace `ip2geotools` vytvořené v programovacím jazyce Python. Aplikaci jsem rozdělil na dílčí části, které je možno znovu použít v jiných programech. Dále jsem vytvořil spustitelný program, který tyto dílčí části spojuje do jednoho většího funkčního celku, čímž jsem demonstroval, že mnou vytvořené moduly do sebe zapadají jako skládačka a že spolu vzájemně spolupracují. V neposlední řadě se musím zmínit také o faktu, že jsem svoji aplikaci dal volně k dispozici třetím stranám pod licencí MIT, zdrojový kód je verzován v Gitu, zveřejněn na GitHubu a také je zanesen do veřejného seznamu balíčku open-source knihoven pro Python zvaného PyPi.

Následně jsem vytvořenou aplikaci `ip2geotools` využil v jednoduchém skriptu, kterým jsem analyzoval vybrané volně dostupné seznamy IP adres, které souvisejí s různými typy kybernetických útoků. Z analýzy vyplývá, že Čína není vždy původcem číslo 1, jak by se mohlo laické veřejnosti na první pohled zdát. Do kybernetických útoků jsou zapojeny i Spojené státy americké, Rusko nebo Indie. Získané poznatky víceméně odpovídají i jiným odborným publikacím.

# Literatura


[1] POESE, Ingmar, Steve UHLIG, Mohamed Ali KAAFAR, Benoit DONNET a Bamba GUEYE. *IP Geolocation Databases: Unreliable?*. ACM SIGCOMM Computer Communication Review [online]. 2011, 41(2), 53- [cit. 2019-04-29]. DOI: 10.1145/1971162.1971171. ISSN 01464833. Dostupné z: `http://portal.acm.org/citation.cfm?doid=1971162.1971171`

[2] PUŽMANOVÁ, Rita. *TCP/IP v kostce*. 2., upr. a rozš. vyd. České Budějovice: Kopp, 2009, 619 s. ISBN 978-80-7232-388-3.

[3] *Proceedings of the 2006 ACM SIGCOMM Internet Measurement Conference: IMC 2006, October 25-27, 2006, Rio de Janeiro, Brazil*. New York, N.Y.: Association for Computing Machinery, c2006. ISBN 1-59593-561-4.

[4] *Comprehensive IP Geolocation Database Downloads | DB-IP. IP Geolocation API & Free Address Database | DB-IP*. [online]. 2019 [cit. 2019-04-29]. Dostupné z: `https://db-ip.com/db/`

[5] *IP to City Lite Free Database Download. IP Geolocation API & Free Address Database | DB-IP*. [online]. 2019 [cit. 2019-04-29]. Dostupné z: `https://db-ip.com/db/download/ip-to-city-lite`

[6] *Find Out About IP Search With Hostip.info. IP Address Lookup Hostip.info*. [online]. [cit. 2019-04-29]. Dostupné z: `http://www.hostip.info/faq.html`

[7] *IP Address Lookup Hostip.info. IP Address Lookup Hostip.info*. [online]. [cit. 2019-04-29]. Dostupné z: `http://www.hostip.info/`

[8] *About - ipstack. ipstack - Free IP Geolocation API*. [online]. [cit. 2019-04-29]. Dostupné z: `https://ipstack.com/about`

[9] *Pricing - ipstack. ipstack - Free IP Geolocation API*. [online]. [cit. 2019-04-29]. Dostupné z: `https://ipstack.com/product`

[10] *GeoLite2 Free Downloadable Databases - MaxMind Developer Site*. [online]. 2012 [cit. 2019-04-29]. Dostupné z: `https://dev.maxmind.com/geoip/geoip2/geolite2/`

[11] *IP2Location LITE Databases | IP2Location LITE*. [online]. 2011 [cit. 2019-04-29]. Dostupné z: `https://lite.ip2location.com/databases`

[12] *GeoIP2 City Database | MaxMind*. [online]. 2012 [cit. 2019-04-29]. Dostupné z: `https://www.maxmind.com/en/geoip2-city/`

[13] *IP Address Geolocation Lookup Demo | IP2Location. IP Address to Identify Geolocation Information | IP2Location*. [online]. 2001 [cit. 2019-04-29]. Dostupné z: https://www.ip2location.com/demo/

[14] *About Us | Neustar. Helping Clients Grow, Guard, and Guide their Businesses | Neustar*. [online]. 2019 [cit. 2019-04-29]. Dostupné z: https://www.home.neustar/about-us

[15] *Get City Details | Geobytes*. [online]. [cit. 2019-04-29]. Dostupné z: http://geobytes.com/get-city-details-api/

[16] *Skyhook | About Us: Precision Location and Actionable Contextual Data. Skyhook | Location Technology and Intelligence*. [online]. 2019 [cit. 2019-04-29]. Dostupné z: https://www.skyhook.com/about-skyhook

[17] *Paid Plans - IPinfo IP Address Geolocation API*. [online]. 2019 [cit. 2019-04-29]. Dostupné z: https://ipinfo.io/pricing

[18] *IP Address Geolocation to trace Country, Region, City, ZIP Code, etc*. [online]. [cit. 2017-12-06]. Dostupné z: https://www.eurekapi.com/

[19] *ipdata: IP Geolocation, Mobile Carrier, Company & Threat Intelligence API*. [online]. [cit. 2019-04-29]. Dostupné z: https://ipdata.co/

[20] KOLOUCH, JUDr. Jan. *Kybernetické útoky*. [online]. Praha [cit. 2019-04-29]. Dostupné z: https://csirt.cesnet.cz/_media/cs/documents/kyberneticke_utoky.pdf

[21] *Nevyžádaná pošta a SPAM | Seznam Nápověda*. [online]. [cit. 2019-04-29]. Dostupné z: https://napoveda.seznam.cz/cz/email/nevyzadana-posta-a-spam/

[22] *Spam - INTERNETEM BEZPEČNĚ. INTERNETEM BEZPEČNĚ - Užívejme internet bezpečnějším způsobem*. [online]. 2018 [cit. 2019-04-29]. Dostupné z: https://www.internetembezpecne.cz/internetem-bezpecne/podvodne-praktiky/spam/

[23] *SPAM. UVT UK - Ústav výpočetní techniky Univerzity Karlovy*. [online]. [cit. 2019-04-29]. Dostupné z: [online].Dostupnéz:http://uvt1.cuni.cz/email/spam/uvod.html

[24] *Jak zabránit spamování a penalizaci webu? - IT Logica. Články a studie - marketing, tvorba webů - IT Logica*. [online]. [cit. 2019-04-29]. Dostupné z: http://blog.it-logica.cz/jak-zabranit-spamovani-penalizace-webu

[25] *Co je malware a jak ho odstranit | Ochrana proti malware | Avast.* [online]. [cit. 2019-04-29]. Dostupné z: https://www.avast.com/cs-cz/c-malware

[26] *Co je malware? | ESET. Malware Protection & Internet Security | ESET.* [online]. [cit. 2019-04-29]. Dostupné z: https://www.eset.com/cz/malware/

[27] *Co je ransomware a jak se ho zbavit | Avast.* [online]. [cit. 2019-04-29]. Dostupné z: https://www.avast.com/cs-cz/c-ransomware

[28] *What is Ransomware, how it attacks and how to prevent | ESET. Malware Protection & Internet Security | ESET.* [online]. [cit. 2019-04-29]. Dostupné z: https://www.eset.com/cz/ransomware/

[29] *About Us: Our Mission | WordPress.org. Blog Tool, Publishing Platform, and CMS — WordPress.* [online]. [cit. 2019-04-29]. Dostupné z: https://wordpress.org/about/

[30] *Security | WordPress.org. Blog Tool, Publishing Platform, and CMS — WordPress.* [online]. [cit. 2019-04-29]. Dostupné z: https://wordpress.org/about/security/

[31] *Common WordPress Attacks and How to Stop Them - WPExplorer. WPExplorer | WordPress Tips, Tutorials, Resources, Themes & Plugins.* [online]. 2019 [cit. 2019-04-29]. Dostupné z: https://www.wpexplorer.com/common-wordpress-attacks/

[32] POLČÁK Libor. *Základní informace o síti Tor.* FIT-TR-2017-01, Brno, 2017 [cit. 2019-04-29]. Dostupné z: https://www.fit.vutbr.cz/research/pubs/index.php.en?file=%2Fpub%2F11513%2Ftr.pdf&id=11513

[33] *Tor Project | History. Tor Project | Anonymity Online.* [online]. [cit. 2019-04-29]. Dostupné z: https://www.torproject.org/about/history/

[34] *Tor Exit Nodes Mapped and Located | HackerTarget.com. 28 Online Vulnerability Scanners & Network Tools | HackerTarget.com.* [online]. 2019 [cit. 2019-04-29]. Dostupné z: https://hackertarget.com/tor-exit-node-visualization/

[35] RAYMOND, Eric Steven. *The Art of UNIX Programming.* Addison-Wesley Professional, 2003, 560 s. Dostupné také z: http://www.catb.org/~esr/writings/taoup/html/

[36] *abc — Abstract Base Classes — Python 3.7.3 documentation.* [online]. [cit. 2019-04-29]. Dostupné z: https://docs.python.org/3/library/abc.html

[37] *Selectors | jQuery API Documentation. jQuery API Documentation.* [online]. [cit. 2019-04-29]. Dostupné z: https://api.jquery.com/category/selectors/

[38] *The MIT License | Open Source Initiative.* [online]. [cit. 2019-04-29]. Dostupné z: https://opensource.org/licenses/MIT

[39] *About · GitHub. The world's leading software development platform · GitHub.* [online]. 2019 [cit. 2019-04-29]. Dostupné z: https://github.com/about

[40] *PyPI – the Python Package Index · PyPI. PyPI – the Python Package Index · PyPI.* [online]. 2019 [cit. 2019-04-29]. Dostupné z: https://pypi.org/

[41] PILGRIM, Mark. *Ponořme se do Python(u) 3: Dive into Python 3.* Praha: CZ.NIC, c2010. CZ.NIC. ISBN 978-80-904248-2-1.

[42] *iX Magazin für professionelle Informationstechnik.* [online]. 2019 [cit. 2019-04-29]. Dostupné z: https://www.heise.de/ix/NiX-Spam-DNSBL-and-blacklist-for-download-499637.html

[43] *nixspam by NiX Spam, spam IPs list, at FireHOL IP Lists.* [online]. [cit. 2019-03-26]. Dostupné z: http://iplists.firehol.org/?ipset=nixspam

[44] *StopForumSpam - FAQ. Stop Forum Spam.* [online]. [cit. 2019-04-29]. Dostupné z: https://www.stopforumspam.com/faq

[45] *stopforumspam_365d by StopForumSpam.com, abuse IPs list, at FireHOL IP Lists.* [online]. [cit. 2019-03-26]. Dostupné z: http://iplists.firehol.org/?ipset=stopforumspam_365d

[46] *MDL.* [online]. 2009 [cit. 2019-04-29]. Dostupné z: http://www.malwaredomainlist.com/

[47] *Downloadable Lists. MDL.* [online]. 2017 [cit. 2019-04-29]. Dostupné z: http://www.malwaredomainlist.com/forums/index.php?topic=3270.0

[48] *malwaredomainlist by MalwareDomainList.com, malware IPs list, at FireHOL IP Lists.* [online]. [cit. 2019-03-26]. Dostupné z: http://iplists.firehol.org/?ipset=malwaredomainlist

[49] *Tracker | Ransomware Tracker.* [online]. 2016 [cit. 2019-04-29]. Dostupné z: https://ransomwaretracker.abuse.ch/tracker/

[50] *ransomware_feed by Abuse.ch, malware IPs list, at FireHOL IP Lists*. [online]. [cit. 2019-03-26]. Dostupné z: http://iplists.firehol.org/?ipset=ransomware_feed

[51] *badips.com | an IP based abuse tracker*. [online]. 2013 [cit. 2019-04-29]. Dostupné z: https://www.badips.com/

[52] *bi_wordpress_2_30d by BadIPs.com, attacks IPs list, at FireHOL IP Lists*. [online]. [cit. 2019-03-26]. Dostupné z: http://iplists.firehol.org/?ipset=bi_wordpress_2_30d

[53] *tor_exits by TorProject.org, anonymizers IPs list, at FireHOL IP Lists*. [online]. [cit. 2019-03-26]. Dostupné z: http://iplists.firehol.org/?ipset=tor_exits

[54] HULTEN, G., J. GOODMAN a R. ROUNTHWAITE. *Filtering spam e-mail on a global scale*. In: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters - WWW Alt. '04 [online]. New York, New York, USA: ACM Press, 2004, 2004, s. 366- [cit. 2019-04-29]. DOI: 10.1145/1013367.1013478. ISBN 1581139128. Dostupné z: http://portal.acm.org/citation.cfm?doid=1013367.1013478, http://www.ra.ethz.ch/CDStore/www2004/docs/2p366.pdf

[55] SHIN, Youngsang, Minaxi GUPTA a Steven MYERS. *Prevalence and mitigation of forum spamming*. In: 2011 Proceedings IEEE INFOCOM [online]. IEEE, 2011, 2011, s. 2309-2317 [cit. 2019-04-29]. DOI: 10.1109/INFCOM.2011.5935048. ISBN 978-1-4244-9919-9. Dostupné z: http://ieeexplore.ieee.org/document/5935048/

[56] MITCHELL, Joshua. *Ransomware Characteristics by Country*. [online]. Medford, 2018 [cit. 2019-04-29]. Dostupné z: http://www.cs.tufts.edu/comp/116/archive/fall2018/jmitchell.pdf

# Seznam symbolů, veličin a zkratek

| | |
|---|---|
| **API** | rozhraní pro programování aplikací – Application Programming Interface |
| **CSS** | kaskádové styly – Cascading Style Sheets |
| **CSV** | hodnoty oddělené čárkami – Comma-Separated Values |
| **DNS** | hiearchický systém doménových jmen – Domain Name System |
| **GPS** | globální polohovací systém – Global Positioning System |
| **HTML** | HyperText Markup Language |
| **HTTP** | Hypertext Transfer Protocol |
| **IANA** | Internet Assigned Numbers Authority |
| **IP** | Internet Protocol |
| **ISP** | poskytovatel internetového připojení – Internet Service Provider |
| **JSON** | JavaScriptový objektový zápis – JavaScript Object Notation |
| **JSONP** | JSON with Padding |
| **KISS** | Zachovej to jednoduché, hlupáku! – Keep It Simple, Stupid! |
| **MAC** | jednoznačný identifikátor síťového zařízení – Media Access Control |
| **OOP** | objektově orientované programování – Object-Oriented Programming |
| **REST** | Representational State Transfer |
| **RTT** | obousměrné zpoždění – Round Trip Time |
| **SSID** | identifikátor bezdrátové sítě Wi-Fi – Service Set Identifier |
| **SSL** | zabezpečený komunikační protokol používaný pro šifrování přenosu informací po síti – Secure Sockets Layer |
| **URL** | jednotná adresa zdroje – Uniform Resource Locator |
| **Wi-Fi** | sada standardů popisujících bezdrátový přenos dat v počítačových sítích – Wireless Fidelity |
| **XML** | rozšiřitelný značkovací jazyk – Extensible Markup Language |

# Seznam příloh

# A Obsah přiloženého CD

- `/` ..... kořenový adresář přiloženého CD
  - `tomas_caha_164249_diplomova_prace.pdf` ..... tato diplomová práce
  - `ip2geotools-master` ..... adresář se zdrojovým kódem aplikace `ip2geotools` (`master` větev zveřejněná na GitHubu)
    - `ip2geotools`
      - `databases`
        - `__init__.py`
        - `commercial.py`
        - `interfaces.py`
        - `noncommercial.py`
      - `__init__.py`
      - `__main__.py`
      - `cli.py`
      - `errors.py`
      - `models.py`
    - `tests`
    - `.gitignore`
    - `CHANGELOG.rst`
    - `LICENSE`
    - `MANIFEST.in`
    - `README.rst`
    - `requirements.txt`
    - `setup.py`
  - `ipanalysis` ..... adresář s analýzou zdrojů kybernetických hrozeb
    - `db` ..... adresář obsahující geolokační databázi použitou k analýze
    - `lists` ..... adresář s vybranými seznamy z FireHOL IP Lists
    - `pages` ..... adresář s detaily k jednotlivým seznamům
    - `results` ..... adresář s výsledky analýzy
    - `ipanalysis.py` ..... skript použitý k hromadné geolokaci IP adres